计算机学院 2006 级硕士研究生

《计算机系统结构》考试题

二OO七年一月七日

学号 _____ 姓名 _____

注意：

1、用中文答题

2、考试期间禁止使用手机、MP3、可编程计算器、电子词典，*一经发现* 按违纪处理

3、考试结束后交回 *考试题* 和 *答卷*，否则不计分

4、考试时间：19:00~21:30

---

1. 名词解释（每个 3 分，共计 30 分）

（1）Branch Delay Slot
（2）Superscale Processor
（3）Globe code Scheduling
（4）Decoupled Architecture
（5）One-bit BPB
（6）Architecture Registers
（7）Home 结点
（8）网络直径
（9）非绑定（nonbinding）
（10）虫孔路由

2. (15 分) What are pipelining hazards? How many types of hazards are there? Give an example of each type and describe how it can be resolved.

3. (12 分) We begin by looking at a simple pipeline, using the pipeline latencies from the table.

Instruction producing result	Instruction using result	Latency in clock cycles
FP ALU op	Another FP ALU	3
FP ALU op	Store double	2
Load double	FP ALU op	1
Load double	Store double	0

And there is one branch slot. For the following code:

```
foo:    ld      f2,0(r1)      ;load X[i]
        multd   f4,f2,f0      ;multiply a*X[i]
        ld      f6,0(r2)      ;load Y[i]
        addd    f6,f4,f6      ;add a*X[i]+Y[i]
        sd      0[r2],f6      ;store Y[i]
        addi    r1,r1,#8      ;increment X index
        addi    r2,r2,#8      ;increment Y index
        sgti    r3,r1,done    ;test if done
        beqz    r3,foo        ;loop if not done
```

(1) Determining each instruction issuing clock and how many cycles per iteration without any scheduling?

(2) Unrolling the loop as many times as necessary to schedule it without stalls and how many cycles per iteration after scheduling?

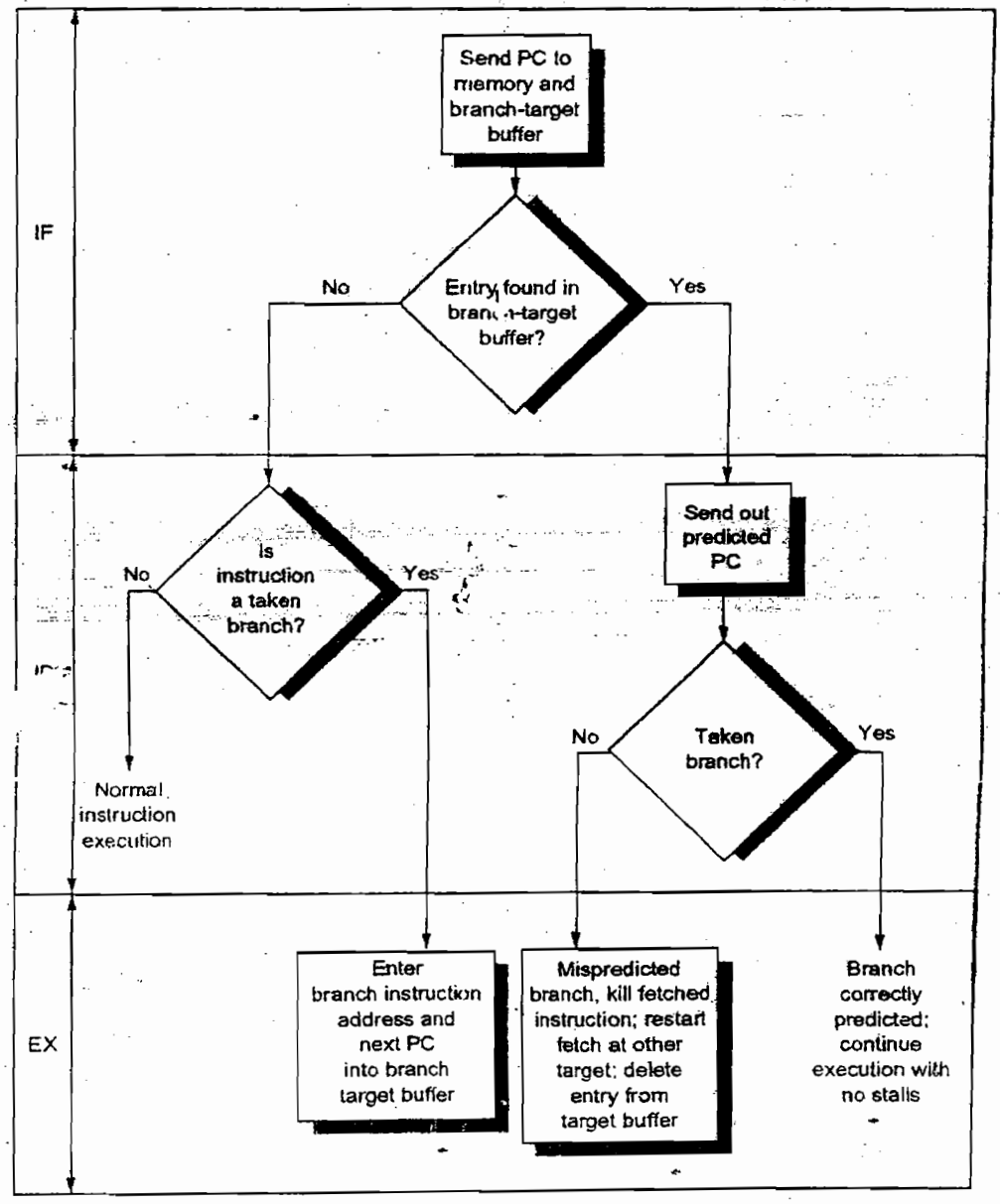
(3) What are the overheads and speedup?

4. (5 分)给出同时多线程（Simultaneous Multithreading）的并行工作原理。

5.（5 分）在标准的栅栏(barrier)同步中，设单个处理器的通过时间(包括更新计数和释放锁)为 C，给出 N 个处理器一起进行一次同步所需要的时间表达式。

6. (13 分)什么是多处理机的相关性(coherency)和一致性(consistency)？给出解决相关性的监听协议的工作原理。

7.（20 分） (1) According to the following figure, explain the steps of a BTB.



(2) Consider a loop that branch behavior is taken 9 times in a row then not taken once. We assume that prediction accuracy is 90% and for instruction in buffer the hit rate in the BTB is 90% (for branches predicted taken). Determine the total branch penalty.

计算机学院2006级硕士研究生

《计算机系统结构》答案

二〇〇七年一月三日

1. 名词解释（每个3分，共计30分）

（1）Branch delay slot: in the delayed branch scheme, the following one instruction cycle of the branch is a ~. 关键：延迟分支策略、下一个时钟周期

（2）Superscale processor: a processor that issue varying numbers of instructions per clock

（3）Globe code scheduling: moving instructions across branches, which is global code scheduling.

(4) Decoupled architecture: A processor with queues to allow slippage of loads and stores with other functional units is called ~.

(5) One-bit branch prediction (BPB): One bit array memory indexed by the lower bits of the branch instruction address contains whether the branch at this address was recently taken or not taken.

(6) Architecture Registers: 是指程序员可以使用的寄存器。

(7) home 结点：存放数据的存储器单元及目录所在的结点

(8) 网络直径：网络中任意两个结点间最短路径的最大值

(9) 非绑定（nonbinding）：预取能返回最新数据值，并且保证对数据实际的存储器访问返回的是最新的数据项

(10) 虫孔路由：把消息包分成小片，片头带目的地址，所有片以不可分离的流水方式通过片缓冲区进行传输路由

2. (15分) What are pipelining hazards? How many types of hazards are there? Give an example of each type and describe how it can be resolved.

答案

- **Pipelining Hazards** prevent next instruction from executing during its designated clock cycle
- There are three types of hazards
  - Structural hazards: HW cannot support this combination of instructions
  - Data hazards: Instruction depends on result of prior instruction still in the pipeline. There are Three Generic Data Hazards
    - Read after write
    - Write after read
    - Write after write
  - Control hazards: Caused by delay between the fetching of instructions and decisions about changes in control flow

Example of Structural hazards: Structural Hazards: One Port Memory for a load and an instruction fetch. It may be resolved by a two port memory.

Example of a Read after write Data hazards:

- add r1,r2,r3
- sub r4,r1,r3

resolved by forwarding

WAR

- I: sub r4,r1,r3
- J: add r1,r2,r3

WAW

- I: sub r1,r4,r3
- J: add r1,r2,r3

The WAR and WAW can be solved by renaming.

Example of the Control hazards is a conditional branch. It can be solved by the delayed branch / branch prediction / speculation etc. scheme.

3. (12 分) We begin by looking at a simple pipeline, using the pipeline latencies from the table.

Instruction producing result	Instruction using result	Latency in clock cycles
FP ALU op	Another FP ALU op	3
FP ALU op	Store double	2
Load double	FP ALU op	1
Load double	Store double	0

And there is one branch slot. For the following cod :

```
foo:    ld      f2,0(r1)      ;load X[i]
        multd   f4,f2,f0      ;multiply a*X[i]
        ld      f6,0(r2)      ;load Y[i]
        addd    f6,f4,f6      ;add a*X[i]+Y[i]
        sd      0[r2],f6      ;store Y[i].
        addi    r1,r1,#8      ;increment X index
        addi    r2,r2,#8      ;increment Y index
        sgti    r3,r1,done    ;test if don
        beqz    r3,foo        ;loop if no  lone
```

(1) Determining each instruction issuing clock and how many cycles per iteration without any scheduling?

(2) Unrolling the loop as many times as necessary to schedule it without stalls and how many cycles per iteration after scheduling?

(3) What are the overheads and speedup?

答案:

(1)(4 分)

```
foo:    ld      f2,0(r1)      1
        multd   f4,f2,f0      3
        ld      f6,0(r2)      4
        addd    f6,f4,f6      7
        sd      0[r2],f6      8
        addi    r1,r1,#8      11
        addi    r2,r2,#8      12
        sgti    r3,r1,done    13
        beqz    r3,foo        14
        (stall)               15
```

It needs 15 cycles per iteration without any scheduling because of the 14 cycles of instruction execution and additional one stall cycle.

(2) (4 分) Unrolling twice, using three more registers, f8, f10, f12, for register renaming:

```
foo:    ld      f2,0(r1)        1
        ld      f8,+8(r1)       2
        multd   f4,f2,f0        3
        multd   f10,f8,f0       4
        ld      f6,0(r2)        5
        ld      f12,+8(r2)      6
        addd    f6,f4,f6        7
        addd    f12,f10,f12     8
        addi    r1,r1,#16       9
        addi    r2,r2,#16       10
        sd      -16[r2],f6      11
        sgti    r3,r1,done      12
        beqz    r3,foo          13
        sd      -8[r2],f12      14
```

It needs 14/2 = 7 cycles per iteration.

(3) (4 分) The overheads are 3 more registers, f8, f10 and f12. Versus the original code, the speedup is 15/7 = 2.1

4 (5 分)给出同时多线程（Simultaneous Multithreading）的并行工作原理。

同时实现指令和线程级的并行，每拍有多个指令槽，可以安排多个线程的多条指令同时流出。

5 (5 分)在标准的栅栏(barrier)同步中，设单个处理器的通过时间(包括更新计数和释放锁)为 C，给出 N 个处理器一起进行一次同步所需要的时间表达式。

答：第一个处理器通过需时间 NC，第二个处理器通过需时间（N-1）C，依次类推，则进行一次同步所需要的时间为：

$NC+(N-1)C+(N-2)C+\cdots+2C+C=N(N+1)C/2$

6 (13 分)什么是多处理机的相关性(coherency)和一致性(consistency)？给出解决相关性的监听协议的工作原理。

答：相关性是指一个数据项的任何读均可得到该数据最近被写的值。一致性是指一个处理器何时读到另一处理器最近更新的内容。

关键是对共享数据写的处理。当某个处理机要写数据时，必须先获得总线的控制权，然后将要作废的数据块地址放在总线上。其它处理机一直监听总线，检测该地址是否存在于它们的 Cache 中。若存在，则作废相应的数据块。获取总线控制权的顺序性保证了写的顺序性，因为两个处理机同时写一个单元时，其中一个处理机必然先获得总线控制权，之后它使另一处理机上对应的拷贝作废，从而保证了写的严格顺序性。

当写 Cache 未命中时，除了将其它处理机上相应的 Cache 数据块作废外，还要从存储器取出该数据块。

监听过程的实现可利用 Cache 中块的标志位。每个块的有效位(valid)使作废机制的实现较为容易。由写作废或其它事件引起的失效处理很简单，只需将该位设置为无效即可。而对于写，我们则希望知道是否别的 Cache 中也有此数据的共享拷贝。因为如果没有别的 Cache 拷贝，则无需将写地址放在写回 Cache 的总线上，这样可降低所用的时间和所需的带宽。在协议中假设操作具有原子性(atomic)，即操作进行过程中不能被打断，例如将写失效的检测、申请总线和接收响应作为一个单独的原子操作。

7.（20 分） (1) According to the following figure, explain the steps of a BTB

(2) Consider a loop that branch behavior is taken 9 times in a row then not taken once. We assume that prediction accuracy is 90% and for instruction in buffer the hit rate in the BTB is 90% (for branches predicted taken). Determine the total branch penalty.

答案：

(1) （10）分支目标缓冲的工作流程以及执行的各个阶段在流水线中的分配：

—如果当前指令的地址与缓冲区中的标示匹配，那么此指令必为分支转移成功指令，且下一条指令的 PC 值在分支目标缓冲的分支目标 PC 域中，因此在本指令指令译码阶段（ID）阶段开始从预测指令的 PC 处开始取下一条指令。

—如果没有匹配而指令进行转移，则分支目标地址会在指令译码阶段（ID）阶段末知道，将其本身的地址和目的地址加入缓冲区中。

—如果在分支目标缓冲中找到了当前指令地址，而指令当前分支不成功，则将此项从分支目标缓冲中删去。

—可以看出，如果是分支指令，并且预测正确则不会有任何延迟；

—如果预测错误，则会耗费一个时钟周期来取错误的指令，并在一个时钟周期后重新取正确指令。

—如果转移指令不在缓冲区中，则将其当成是不成功的分支指令处理，耗费延迟的大小取决于指令是否转移成功。

—分支延迟表如下（用于下一步计算）

Instruction in BTB	Prediction	Actual branch	Penalty cycle
yes	T	T	0
yes	T	NT	2
no		T	2
no		NT	0

(2) （10）Probability (branch in buffer, but not taken)

=Percent buffer hit rate × Percent incorrect predictions

=90% × 10%

=0.09

Probability (branch not in buffer, but actually taken) = 10%

Branch penalty = (0.09 + 0.10) × 2 = 0.38

计算机学院 2004 级硕士研究生

《计算机系统结构》考试题

二〇〇五年一月五日

注意：

1、可以用中文答题

2、考试结束后交回考题

一、名词解释（每个 3 分，共计 30 分）

（1）Out of order issue

（2）Reorder buffer

（3）Loop level parallelism

（4）Branch delay slot

（5）Software pipeline

（6）Structure hazard

（7）home 结点

（8）网络直径

（9）等分带宽

（10）虚拟自适应

2. (10) Tell the advantages and disadvantages of dynamic scheduling.

3. (10) Consider the following code snippet:

```
1:   x←x+1
2:   if x>y then goto 10
3:   goto 20
     ...
10:  w←z
```

in speculative execution, these instructions would be reorder as follows.

```
1:   x←x+1
10:  w←z
2:   if x>y then goto 10
3:   goto 20
```

Which is the speculative instruction and how it executes correct?

4. (10) State the data dependencies, data anti-dependencies, and data output dependencies in following code segment. For each dependency, give the statements involved, the type of dependency, and the dependent variable.

```
1:   A←B+C
2:   D←E+F
3:   G←A+D
4:   A←D+E
```

5. (15 分) 本题中与浮点数据操作有关的的延迟关系如下：

产生结果的指令	使用结果的指令	延迟
浮点算术运算	另一个浮点算术运算	3
浮点算术运算	双精度存	2
双精度取	浮点算术运算	1
双精度取	双精度存	0

整数操作之间以及与其它所有浮点操作之间的延迟为 0，转移指令的延迟为 0。对于公式 SAXPY：

$$Y = a \times X + Y$$

假设向量 X 和 Y 的长度足够长，它的汇编为：

```
foo:  ld     f2,0(r1)      ;load X[i]
      stall
      multd  f4,f2,f0      ;multiply a*X[i]
      ld     f6,0(r2)      ;load Y[i]
      stall
      addd   f6,f4,f6      ;add a*X[i]+Y[i]
      sd     0(r2),f6      ;store Y[i]
      addi   r1,r1,#8      ;increment X index
      addi   r2,r2,#8      ;increment Y index
      sgti   r3,r1,done    ;test if done
      beqz   r3,foo        ;loop if not done
```

(1) 采用标准的单流水线，求出这个原始 SAXPY 循环一次所需要的时间，并求出其中空转(sta
期数。

(2) 如果系统中的功能部件数量不受限制，展开循环 4 遍，经过优化，使 SAXPY 循环处理时间

6. (5)给出同时多线程（Simultaneous Multithreading）的并行工作原理。

7. (5 分)大规模机器的同步有哪些软件和硬件支持方法？

8. (15 分)什么是多处理机的相关性(coherency)和一致性(consistency)？给出解决相关性的目录协
工作原理及状态变迁。

计算机学院2003级硕士研究生

《计算机系统结构》考试题

二〇〇四年一月八日

注意：

1、可以用中文答题

2、考试结束后交回考题

---

1. 名词解释（每个3分，共计30分）

（1）Principle of locality

（2）Relative MIPS

（3）Byte aligned memory access

（4）Branch delay slot

（5）Loop level parallelism

（6）NUMA机器

（7）非绑定（nonbinding）

（8）网络直径

（9）等分带宽

（10）虚拟自适应

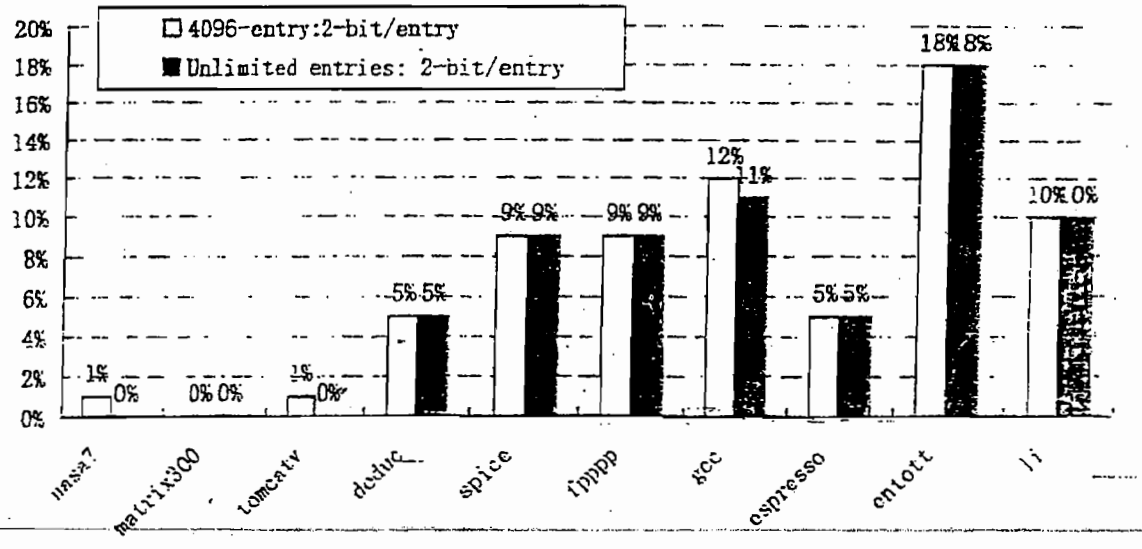
2. (5)大规模机器的同步有哪些软件和硬件支持方法？

3. (5)设互联网络的输入为 $x=(x_{k-1},\cdots,x_1,x_0)$，给出在均匀混洗和超立方体两种情况下网络的输出 $y=f(x)$。

4. (5)给出同时多线程（Simultaneous Multithreading）的并行工作原理。

5. (10)Expressing the basic idea of Tomasulo's scheme.

6. (10) What do you learn from the following figure.



Prediction accuracy of a 4096-entry two-bit prediction buffer versus an infinite buffer for the SPEC89 benchmarks

2002 研究生体系结构考试题

1. 名词解释（每个名词3分，共30分）

拥有者（owner）	虚拟自适应
网络直径	等分带宽
虫孔路由	Structure hazard
Instruction window	Globe code scheduling
Branch prediction buffer	Data flow

2. （5分）为什么采用松弛一致性模型(relaxed consistency model)的机器可以提高性能？
3. （5分）给出同时多线程（Simultaneous Multithreading）的并行工作原理。
4. （5分）大规模机器的同步有哪些软件和硬件的支持方法？
5. （15分）一台超节点的分布共享多处理机，节点内部运行监听协议，节点之间运行目录协议，试述系统解决相关性的工作原理。

6. （10分）Consider the following code snippet：

```
1: x = x+1
2: if x>y then goto 10
3: goto 20
...
10: w = z
```

In speculative execution, these instructions would be reorder as follows.

```
1: x = x+1
10: w = z
2: if x>y then goto 10
3: goto 20
```

Which is the speculative instruction and how it executes correct ?

7. （10分）State the data dependencies, data anti-dependencies, and data output dependencies in the following code segment. For each dependency, give the statements involved, the type of dependency and the dependent variable.

```
1: A = B + C
2: D = E + F
3: G = A + D
4: A = D + E
```

Statements involved	type of dep	depend variable
1, 3	RAW	A
1, 4	WAW	A
2, 3	RAW	D
2, 4	RAW	D
3, 4	WAR	A

8. （10分）Consider the execution of a program of 15,000 instructions by a linear pipeline processor with a clock rate of 25 MHz. Assume that the instruction pipeline has five stages that one instruction is issued per clock cycle. The penalty due to branching is ignored in (a), but in (b).

(a) Calculate the speedup factor in using this pipeline to execute the program compared to the use of an equivalent nonpipeline processor with an equal amount of flow-through delay.

(b) What are the efficiency and throughput of this pipelined processor, if 25% additional cycles are needed to execute the same code, if the branching effects are included ?

9. （10分）Tell the advantages and disadvantages of dynamic scheduling.

《高级体系结构》2000考试题

1、名词解释：(中文作答 5'/题；英文作答 6'/题)

Instruction level parallelism

Trace compaction

把一大串路径上的指令压缩成少数几条长指令并同

Branch prediction buffer

尽可能早的调度.

Structure dependences

Reorder buffer

2、(20') Latencies of FP instructions shown in the table.

Inst. Producing result	Inst. Using result	Latency in clock cycles
FP ALU op.	Another FP ALU op.	3
FP ALU op.	Store double	2
Load double	FP ALU op.	1
Load double	Store double	0

There is a one clock branch delay that can be used as a branch slot. For the following codes:

```
Loop: LD    F0, 0 (R1)      LD
      ADDD  F4, F0 ,F2      ADD
      SD    0 (R1), F4      SD
      SUBI  R1, R1, #8      LD
      BNEZ  R1, Loop.
```

10

- ✓ Calculating the clock cycles of one loop WITHOUT instruction schedule.
- ✓ Calculating the clock of one loop WITH instruction schedule. 6
- ✓ Show the scheduled program in four copies of loop body that not have any latency.
- ✓ Listing the cycles per element in previous and calculating speedups each.

3、(15')Full the following blanks:

技术	主要克服的停顿
Loop unrolling	控制相关
Pipeline scheduling	编写编译
Dynamic scheduling	所有数据
Branch prediction	控制
Speculation	所有
Multiple issuing	

4、(15')The Latencies of a branch targer buffer BTB

指令在 BTB 中	预测结果	实际动作	延迟周期
是	成功	成功	0
是	成功	不成功	2
不是		成功	2

According to the following assumptions, calculating the latency of a branch.

- 预测准确率 90%
- 缓冲区命中率 90%
- 假设分之转移成功的比例为 60%

90% × 10% × 2
+ 10% × 0.6 × 2.
0.12 + = 0.3
0.18

5、(20') To following loop:

```
Loop: LD     F0, 0 (R1)
      ADDD   F4, F0, F2
      SD     0 (R1), F4
      SUBI   R1, R1, #8
      BNEZ   Loop
```

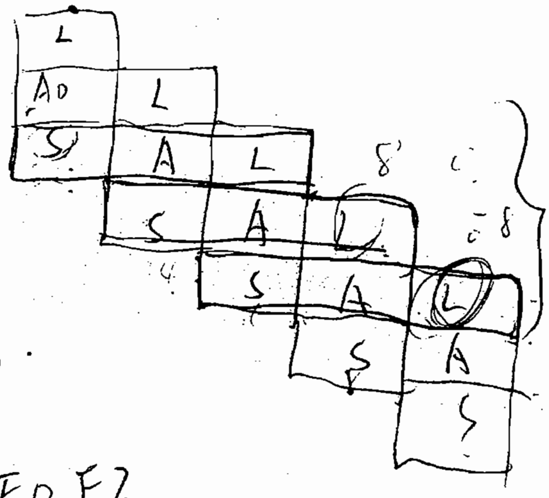
10

1) Calculating the clock cycles of the loop; (5')
2) Show a software-pipelined version of the loop and calculating the cycles of a loop: (5')
3) Supposing ADDD is 6 cycles. How to avoid the stalls? Show the code. (10')

??

```
SD    16(R1), F4
ADDD  F4, F0, F2.
LD    F0, 0(R1)
SUBI  R1, R1, #8.
BNEZ.
```

7.



```
SD   32(R1), F4
ADD  F4, F0, F2,
SD   24(R1) F5,
ADD  F5, F6, F2,
```

5.

LD F0.

《高级体系结构》1999考试题

1、解释名词 （30’）

指令级并行：指令序列中存在着潜在并行性称为指令级并行。

This potential overlap among instructions is called *instruction-level parallelism* (ILP) since the instructions can be evaluated in parallel.

指令调度：通过改变指令在程序中的位置，将相关指令之间的距离加大到不小于指令执行延迟的时钟数，即将相关指令转化为实际无关的指令。

循环展开：通过展开循环体若干次，将循环级并行性转化为指令级并行

数据相关：j 指令使用 I 指令的结果，即写后读相关。

名相关：两条指令使用相同的名，但他们之间没有数据流，称之为名相关。

A、输出相关：两条指令写相同的名，即写后写相关。

B、反相关：指令 I 先执行，指令 j 写的名是 I 读的名，即读后写相关。

动态调度：通过硬件重新安排指令的执行循序，来调整相关指令实际执行时的关系，减少处理机空转。

important technique, in which the hardware rearranges the instruction execution to reduce the stalls while maintaining data flow and exception behavior.

静态调度：通过编译器调度导致阻塞相关的指令，减少处理机空转。

记分牌：在指令动态调度中，通过相应电路检测到指令运行所需的资源满足并没有数据相关，就允许指令乱序执行同时记录下这些指令的运行状态。

Scoreboarding is a technique for allowing instructions to execute out of order when there are sufficient resources and no data dependences.

寄存器从命名：使用大量的虚拟寄存器来替代源代码中的寄存器。

Tomasulo 算法：通过动态调度、寄存器重命名、动态存储器地址判别技术来解决 waw 相关和 war 相关。

保留站：保存等待流出和正在流出指令所需的操作数。

公共数据总线：计算结果通过相关专用通路直接从功能部件到对应的保留站进行缓冲，相关专用通路通过一条公用数据总线来实现。

分支预测缓冲(BPB)：使用一片缓冲区记录最近一次或几次的分支历史，用分支指令地址的低位来索引，缓冲的存储区为 1 位的分支历史记录位，称为预测位。

分支目标缓冲（BTB）：将分支成功的分支指令和它的分支目标地址都放到一个缓冲区内保存起来，缓冲区以分支指令的地址作为标识。

推断执行：处理机还未判断指令是否能执行之前就提前执行，即克服了控制相关。

再定序缓冲：在推断执行中为了实现乱续执行但顺序确认，故加入指令确认阶段需要一套额外的硬件缓冲，来保存那些执行完但未经过确认的指令及其结果．这个硬件的缓冲称为再定序缓冲，同时还用他来在推断执行的指令之间传送结果。

超标量（superscalar）：即每个时钟周期流出的指令数是不定，它既可以通过编译器静态调度，也可以通过记分牌或 tomasulo 算法等动态调度实现多指令流出。

超流水（superpileling）：是将每个功能部件进一步流水化，使得一个功能部件在一拍中可以处理多条指令。

超长指令字（VLIW）：每个时钟周期流出的指令数是固定的，他们构成一条长指令，或者是一个混合的指令包。

Trace compaction: Once a trace is selected is selected ,the second process called trace compaction. It attempts to move operations as early as it can in a sequence(trace), packing the operations into a few wide instructions as possible.

集中式共享存储器（CSMA）：处理器数目较小，可通过大容量的 cache 和总线使各处理器共享一个单独的集中式存储器。

分布式共享存储器(DSM,distributed shared-memory)或可缩放共享存储器(SSM，scalable shared-memory）体系结构：物理上分离的多个存储器可作为一个逻辑上共享的存储空间进行编址,这样一个处理器可以访问任何一个其他的局部存储器.

通讯延迟：发送开销+跨越时间+传输时间+接收时间

计算/通讯比：是衡量并行程序性能的尺度。

超结点：每个结点内还可能包含较小数目（2-8）的处理器，这些处理器之间可采用另一种技术（例如通过总线）互连形成簇(cluster)，这样形成的结点叫做超结点。

迁移：是把远程共享数据项的拷贝放在本处理器的一个局部 Cache 中使用，从而降低了对远程共享数据的访问延迟。

复制：是把多个处理器需要同时读取的共享数据项的拷贝放在各自局部 Cache 中使用，复不仅降低了访存的延迟，也减少了对共享数据的访问产生的冲突。

目录（directory）：物理存储器中共享数据块的状态及相关信息均被保存在一个称为目录的地方。

监听(snooping)：每个 Cache 除了包含物理存储器中块的数据拷贝之外，也保存着各个块的共享状态信息。Cache 通常连在共享存储器的总线上，各个 Cache 控制器通过监听总线来判断它们是否有总线上请求的数据块。

写作废（write invalidate）协议：在一个处理器写某个数据项之前保证它对此数据项有唯一的访问权。

写更新协议：当一个处理器写某数据项式，通过广播使其它 cache 中所有对应的该数据项拷贝进行更新。原子性(atomic)，即操作进行过程中不能被打断，例如将写失效的检测、申请总线和接收响应作为一个单独的原子操作。

互联网络：是将集中式系统或分布式系统中的节点连接起来所构成的网络。

结点度：与结点相连接的边的数目称为结点的度。

网络直径：网络中任意两个结点间最短路径长度的最大值称为网络直径。虚拟自适应：将一个物理通道分成几个虚拟的通道，根据后续各虚拟通道的忙闲情况自适应选择后续通道。拥有者（owner）：拥有唯一的 Cache 块拷贝的处理器通常称为这个 Cache 块的拥有者(owner)，处理器的写操作使自己成为对应 Cache 块的拥有者。

等分带宽：在将某一网络切成相等两半的各种切法中，沿切口的最小通道边数称为通道等分宽度 b (channel bisection width)。

路由（routing）：在网络通信中对路径的选择与指定。

静态网络：由点和点直接相连而成，这种连接方式在程序执行过程中不会改变。

动态网络：是用开关通道实现的，它可动态地改变结构，使其与用户程序中通信要求匹配。

虫孔路由：虫蚀（wormhole） 把包进一步分成小片，硬件路由器有片缓冲区，同一个包中所有片象不可分离的同伴一样，以流水方式顺序传送。只有片头包含目标地址，所有片必须跟随片头。

循环的相关距离

2、（5’）对多线程 MPP 系统进行分析考虑的主要参数以及多线程的切换策略。

答：主要用到四个参数：（1）时延：即远程通信时延；（2）线程个数：线程用一个现场表示，现场由程序计数器、寄存器组和所要求的现场状态字构成；（3）现场切换开销；（4）两次切换的间隔。

切换策略：（1）在 Cache 失效时（2）每次加载时（3）执行每条指令时（相关性小，并行好）（4）执行指令块时

3、（5’）简单比较动态网络中总线、多级网络、交叉开关的特点。

答：构成动态网络的总线、多级网络、交叉开关中，总线的造价最低，但其缺点是每台处理器可用的带宽较窄。总线所存在的另一个问题是容易产生故障。有些容错系统，如用于事务处理的 Tandem 多处理机等，常采用双总线以防止系统产生简单的故障。

由于交叉开关的硬件复杂性以 n2 上升，所以其造价最为昂贵。但是，交叉开关的带宽和路由性能最好。如果网络的规模较小，它是一种理想的选择。

多级网络则是两个极端之间的折衷。它的主要优点在于采用模块结构，因而可扩展性较好。然而，其时延随网络的级数而上升。另外，由于增加了连线和开关复杂性，价格也是一种限制因素。

4、（5’）在标准的栅栏（barrier）同步中，设耽搁处理器的通过时间（包括更新技术和释放锁）为 C，给出 N 个处理器一起进行一次同步所需的时间表达式。

5、（5’）指令的延迟如表所示：

Inst. Producing result	Inst. Using result	Latency in clock cycles
FP ALU op.	Another FP ALU op.	3
FP ALU op.	Store double	2
Load double	FP ALU op.	1
Load double	Store double	0

并且有一个周期的分支延迟（Branch Delay），可用作转移槽。对于下面的指令段：

```
Loop: LD     F0, 0 (R1)
      ADDD   F4, F0 ,F2
      SD     0 (R1), F4
      SUBI   R1, R1, #8
      BNEZ   R1, Loop
```

1) 求出指令没有调度的情况下，一次循环的时间；

answer:

```
LOOP: LD   F0,0(R1)          1
      STALL                  2
      ADDD F4, F0 ,F2        3
      STALL                  4
      STALL                  5
      SD     0 (R1), F4      6
      SUBI   R1, R1, #8      7
      BNEZ   R1,LOOP         8
      STALL                  9
```

2) 对于指令进行调度，求出调度后一次循环的时间；

ANSWER: **Loop: LD F0, 0 (R1)** **1**

STALL	2
ADDD F4, F0 ,F2	3
SUBI R1, R1, #8	4
BNEZ R1, Loop	5
SD 8(R1), F4	6

6、（10’）填写下面的表格

技术	主要克服的停顿
循环展开	控制相关
基本流水线调度	数据先写后读相关
指令动态调度	各种数据相关
分支预测	控制相关
推测（Speculation）	所有数据、控制相关
多指令流出	提高理想 CPI

7、（10’）给出四种松弛（relaxed）一致性模型的特点及其实现上所需要的硬件支持措施。

答：第一个模型是松弛写和读(对不同的地址)的顺序，即取消 W→R 顺序，这种模型采用写缓存，并提供读的旁路机制，从而允许处理机在其写的操作被所有的别的处理机看到之前就继续运行读；如果松弛 W→W，允许非冲突写隐含地乱序进行，则成为部分存序(partial store ording PSO)模型，从实现的角度看，可以使写流水化或重叠，而不是强制一个操作必须在另一个之前结束，对同步操作仍需将写操作挂起，因为它引起写防护；松弛的模型的第三类是取消 R→R，R→W，W→R 和 W→W，这种模型称为弱排序(Weak ordering)；进一步弱化的模型称为释放一致性(release consistency)模型，这种模型区分同步操作中的访问一个共享变量的获取操作 SA 和将对象释放允许别的处理机获取访问权的释放操作 SR。

8、（10’）什么是多处理机的相关性（coherency）和一致性（consistency）？给出解决相关性的监听协议的工作原理。

答：如果对某个数据项的任何读操作均可得到其最新写入的值，则认为这个存储系统是一致的。包括了两个不同方面：

相关性：返回给读操作的是什么值(what, Coherence)；

一致性：什么时候才能将已写入的值返回给读操作(when, Consistence)。

1．处理器 P 对 X 进行一次写之后又对 X 进行读，读和写之间没有其它处理器对 X 进行写，则读的返回值总是写进的值。

2．一个处理器对 X 进行写之后，另一处理器对 X 进行读，读和写之间无其它写，则读 X 的返回值应为写进的值。

3．对同一单元的写是顺序化的，即任意两个处理机对同一单元的两次写，从所有处理器看来顺序都应是相同的。

9、（10’）采用记分牌算法，假设浮点流水线中的执行的延迟如下：加法需 2 个时钟周期，乘法需 10 个时钟周期，除法需 40 个时钟周期，代码段和起始状态如下：

指令	指令状态表			
	IS	RO	EX	WR
LD F6, 34 (R2)	√	√	√	√
LD F2, 45 (R3)	√	√	√	
MULTD F0, F2, F4	√			
SUBD F8, F6, F2	√			

DIVD F10, F0, F6	V				
ADDD F6, F8, F2					

部件名称	功能部件状态表								
	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
整数	yes	LD	F2	R3				no	
乘法	yes	MULTD	F0	F2	F4	整数		no	yes
加法	yes	SUBD	F8	F6	F2		整数	yes	no
除法	yes	DIVD	F10	F0	F6	乘法		no	yes

	结果寄存器状态表							
	F0	F2	F4	F6	F8	F10	…	F30
部件名称	乘法	整数			加法	除法		

写出 MULTD 准备写结果之前的记分排状态。

指令	指令状态表			
	IS	RO	EX	WR
LD F6, 34 (R2)	V	V	V	V
LD F2, 45 (R3)	V	V	V	V
MULTD F0, F2, F4	V	V	V	
SUBD F8, F6, F2	V	V	V	V
DIVD F10, F0, F6	V			
ADDD F6, F8, F2	V	V	v	

部件名称	功能部件状态表								
	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
整数	No								
乘法	Yes	Multd	F0	F2	F4			No	No
加法	Yes	Addd	F6	F8	F2			No	No
除法	Yes	divd	F10	F0	F6	乘法		No	Yes

	结果寄存器状态表							
	F0	F2	F4	F6	F8	F10	…	F30
部件名称	乘法			加法		乘法		

10、（10’） 对于分支目标缓冲，按表中给出的分支转移地延迟和下面的假设，计算分支转移总的延迟。

- 预测准确率 90%
- 缓冲区命中率 90%
- 假设分之转移成功的比例为 60%

指令在 BTB 中	预测结果	实际动作	延迟周期
是	成功	成功	0
是	成功	不成功	2
不是		成功	2

Answer: 分支延迟＝（90%*10%*2）+(10%*60%*2)=0.3

《计算机体系结构》

加速比 = (1-可改进比例) + 可改进比例 / 部件加速比 《计算机体系结构》

附录 A Pipeline

1 流水线技术：并行性分析①

实现多条指令重叠执行的技术。它利用了执行指令所需操作之间存在的并行性。

2 流水线中的 Hazard：

Structural Hazard：如果某些指令组合在流水线中执行时，产生资源冲突，称该流水线有~。

Data Hazard：指令在流水线中执行时，流水线有可能改变指令读/写操作的顺序，似的读/写操作顺序不同于它们在非流水实现时的顺序，这将导致~。

Branch Hazard：（略）

第三章 指令级并行和动态调度

并行性：

1 指令级并行：

研究的是一个基本程序块中指令序列之间存在的并行性。

2 指令的调度：

通过改变指令在程序中的位置，将相关指令之间的距离加大到不小于指令执行延迟的时钟数，这样就可以将相关指令转化为实际上无关的指令。

循环展开：通过展开循环体若干次，将循环级并行性转化为指令级并行性的技术。

程序基本块范围内的相关：

3 数据相关：

~~后续指令使用前面指令执行的结果~~ RAW 写后读相关（RAW）。

4 名相关：

两条指令使用相同的名（寄存器），但他们之间没有数据流，这种相关称为名相关。包括：

反相关：指令 i 先于指令 j 执行，指令 j 写的寄存器是指令 i 读的寄存器，WAR 读后写相关(WAR)

输出相关：两条指令写相同的名，WAW 写后写(WAW)相关。

寄存器重命名：使用大批的虚拟寄存器来代替源代码中的寄存器来消除名相关。

5 控制相关：

由分支指令引起的相关，需要根据分支指令执行的结果来确定后续指令的执行顺序。

6 动态调度：

通过硬件对新安排指令的执行顺序，来调整相关指令实际执行时的关系，减少处理器空转。

7 静态调度：通过编译器调度导致阻塞的相关指令，减少处理器空转。

8 记分牌：

在指令的动态调度中，检测指令运行所需要的资源是否可用，有没有数据相关，若资源可用，且没有数据相关，就允许指令乱序执行并记录指令的运行状态。

~~记分牌技术的性能限制因素：~~

1）程序指令中本身可开发的并行性 2）~~记分牌容量~~

3）功能部件的数目和种类 4）~~反相关和输出相关~~

动态调度优点①可以处理很多编译旧无法处理的相关 ②简化的编译旧 ③在不同的流水线上运行相同的代码不需要重新编译.
缺点① 极大的增加了硬件的复杂性.

9 Tomasulo 算法：

通过动态调度，寄存器重命名，动态存储器地址判别来解决 WAW 和 WAR 冲突。

保留站：用来实现寄存器重命名，其作用是保存等待流处和已经流出指令所需的操作数。

公共数据总线（CDB）：计算结果通过相关专用通路，从功能部件进入保留站进行缓冲，而不一定写到寄存器，这一相关专用通路就是公共数据总线（CDB）。

Tomasulo 算法的优缺点：

优点：1）分布的阻塞检测逻辑机制　2）消除了数据 WAW 和 WAR 相关导致的阻塞。

缺点：1）硬件结构复杂　2）异常处理不精确

3）性能还受限于公共数据总线，增加总线的数量，会增加缓冲和保留站接口硬件。

10 动态分支预测：

分支预测缓冲：使用一片缓冲区来记录最近一次或几次的分支历史，用分支指令地址的低位来索引，缓冲的存储区为 1 位的分支历史记录位，称为预测位。

分支目标缓冲：将分支成功的指令的地址和它的分支目标地址都存放在一个缓冲区中保存起来，缓冲区以分支指令地址为标示。这个缓冲区就是分支目标缓冲。（计算题）

11 基于硬件的推断执行：

推断执行：允许处理器在还未判断指令是否执行之前就提前执行。其结合了三种思想：动态的分支预测来决定执行哪条指令，在控制相关消除之前指令推断执行，对基本块采用动态调度。实现推断的关键思想是允许指令乱序执行但顺序确认。

再定序缓冲：用来保存指令执行完毕到指令得到确认之间的所有指令及其结果，同时在推断执行的指令之间传递结果。

12 多指令流出：

超标量：每个时钟周期流出的指令数不定，可以通过编译器静态调度，也可以通过记分牌或 Tomasulo 算法进行动态调度。

VLIW：每个时钟周期流出的指令是固定的，它们构成一条长指令或一个混合的指令包。
硬件较 super scalo 简单，但是执行流不灵活，需要大量 RS

13 Instruction Window：

The set of instruction examined for simultaneous execution is Instruction Window 。

第四章 软件技术开发指令级并行

1 Loop Unrolling：

通过展开循环体若干次，将循环级并行性转化为指令级并行性的技术。它可以消除分支，使得可以对来自不同循环体中的指令同时进行调度。同时(关键)能够减小CPI.

2 指令的调度：

通过改变指令在程序中的位置，将相关指令之间的距离加大到不小于指令执行延迟的时钟数，这样就可以将相关指令转化为实际上无关的指令。

3 静态分支预测：

用于分支行为在编译时间可以准确预测的处理器中　见附录 A
配置文件：分支指令的执行通常双峰分布

4 VLIW：　一个具体的分支或者总被选中，或者总不被选中

每个时钟周期流出的指令是固定的，它们构成一条长指令或一个混合的指令包。

VLIW 的缺陷：

1）代码尺寸增加：为了在直线式代码段中产生足够的操作，需要多次进行循环展开从而增加了代码的尺寸。当指令不满时，未被使用的功能部件对应的指令部分为空。

2）Lockstep 操作的限制：由于没有冲突检测硬件，且所有操作必须同步进行，任何一个功能部件的停顿都会引起整个处理器的停顿。

3）代码的兼容性

5 Loop-carried dependence：

对一个循环体中的数据的访问，依赖于较早循环体产生的数据

6 软件流水：

对循环进行重组，似的循环体由来自原先不同循环体中的指令构成。

通过从不同循环体中选择指令，相关计算被隔离，似的展开的循环可以无停顿的调度。其相对于循环展开的优点是：占用较少的代码空间

步骤：symbolically unroll the loop，then,select instructions from each iteration。

7 全局代码调度：（Global code scheduling）

跨越分支移动指令，将带有内部控制结构的代码段压缩成最短序列，并保持其数据相关和控制相关。全局代码调度可以通过移动代码减少非循环条件分支。

关键路径：一个代码段可能的最短序列。

8 路径调度：(Trace scheduling)

通过增加执行频率低的分支的开销，来简化代码调度。步骤：

1） Trace selection：在多个基本块中，找到一个可能的序列，使多个基本块中的操作可以用少数几条指令来实现。

2） Trace compactation：目的是将路径中的操作尽可能前移，并将所有操作压缩成尽可能少的几条宽指令。

9 Conditional instruction：

当分支指令的行为在编译时间不能准确预测时，采用条件指令可以消除分支，将控制相关转化成数据相关。

第六章 多处理机

1 集中式共享存储器结构：

2 分布共享存储器结构

3 通信机制：

共享存储器机器通过 Load，Store 指令，多地址空间的机器通过消息传递来进行。

3 机制的性能指标：通信带宽，通信延迟，通信延迟的隐藏。

4 不同通信机制的优缺点

5 并行处理面临的挑战：(1)程序中有限的并行性 (2)相对较高的通信开销。

6 并行程序的计算/通信比率

反映的是并行程序的性能。随着处理器加大个

7 集中共享存储器体系结构：

10. 带硬件支持的编译四方法

支持对共享数据和私有数据的 Cache 缓存。共享数据进缓存引发的问题：Cache coherence.

Cache coherence：

如果对某个数据项的任何读操作均可得到其最新写入的值，则，认为这个存储器是 coherent

包含两层意思：Coherence ：返回给读操作的是什么值，

Consistency：什么时候才能将写入的值返回给读操作。

1）处理器 P 对 X 进行一次写之后，又对 X 进行读，读和写之间没有其它处理器对 X 进行写，则，读的返回值总是写进的值。

2）一个处理器对 X 进行写之后，另一个处理器对 X 进行读，读和写之间无其它写，则，读的返回值应是写进的值

3）对同一个单元的写是顺序化的，即任意两个处理器对同一单元的两次写，从所有处理器的角度看来都应该是相同的。写顺序化

8 实现一致性的基本方案：

共享数据的迁移：把远程共享数据项拷贝到本处理器局部的 Cache 中使用，从而减低了对远程共享数据项的访问延迟。

共享数据的复制：把多个处理器需要同时读取的共享数据项的拷贝放在各自局部 Cache 中使用，复制不仅降低了访存的延迟，也减少了访问共享数据所产生的冲突。

Cache 块的状态有三种：

共享：在一个或多个处理器上有这个块的拷贝，且在主存中的值是最新值。]

未缓冲：所有处理器的 Cache 都没有此块的拷贝。

专有：仅有一个处理器上有此块的拷贝，且已对此块进行了写操作，而主存的拷贝仍是旧的。这个处理器称为此块的拥有者。处理器的写操作使自己成为对应 Cache 块的拥有者。

实现 Cache 一致性的关键是跟踪共享数据块的状态。目前有两种协议，它们分别采用了不同的共享数据状态跟踪技术：　只适用于小规模的处理器组织。

1）　监听：　关键是利用总线进行作废操作。当某个处理器进行写数据时，必须先获得总线的控制权，然后将要作废的数据块地址放在总线上。其它处理器一直在监听总线，它们检测该地址所对应的数据块是否在它们的 Cache 中，若在，则作废相应的数据块。

拥有者：唯一拥有一个 Cache 块拷贝的处理器称该 Cache 块的拥有者

2）目录：物理存储器中共享数据块的状态及相关的控制信息均保存在一个称为目录的地方。目录协议的基本特点是：在每个结点内增加了目录存储器用于存放目录，存储器的每一块在目录中对应有一项，每个目录项主要有“状态”和“位向量”两种成分。“状态”描述该目录对应存储器块的状态，“位向量”有 N 位，其中每一位对应于一个处理器（的局部 Cache），用于指出该 Cache 中有无该存储器块的拷贝。当处理器对某一块进行写操作时，只要根据位向量通知具有相应拷贝的处理器进行作废操作。

局部结点：产生请求的结点。

宿主结点：存放有存储器块和对应地址目录项的结点。

远程结点：拥有 Cache 块拷贝的结点（shared or exclusive）

非阻塞。~~读取使数据值在处理器更新，并能保证一旦有访问这个数据时：~~

读取能返回最新的数据〈IV〉值，并且保证对数据是否从存储器访问返回的是最新的数据值。

注：写作废协议：一个处理器在写一个数据项之前，保证它对该数据项拥有唯一的访问权，而别的处理器中的拷贝都作废。对应这种方法的协议称为写作废协议。

写更新协议：当一个处理器写某个数据项时，通过广播使其他 Cache 中所对应的该数据项第二拷贝进行更新。

写作废协议和写更新协议性能上的差别：

1）对一个数据块进行写时，作废或更新的次数

2）作废或更新的操作对象

3）一个处理器写到另一个处理器读之间的延时

9 同步：

在消息传递机制中，如果请求处理器在发送一个请求后，一直要等到应答结果才能继续执行，这种消息传递称同步。

同步机制通常是基于硬件提供的有关同步指令，通过用户级软件例程来建立的。

10 栅栏同步：（计算题）

栅栏强制所有到达该栅栏的进程进行等待，直到全部的进程到达栅栏，然后释放全部进程，从而形成同步。

栅栏同步的实现是用两个旋转锁：一个用来记录到达栅栏的进程数，另一个用来封锁进程直到最后一个进程到达栅栏。

栅栏同步中，总线事务处理数与需要进行同步的进程数之间的关系为：

注：

在锁同步中，同时竞争锁的进程数目与总线事务处理数之间的关系为：

11 为什么采用松弛一致性模型的机器可以提高性能？给出四种松弛一致性模型的特点及其实现上所需的硬件支持措施。P607

12 多线程：允许多线程以重叠的方式共享一个处理器中的功能部件。

细粒度多线程：可以在线程的每条指令上切换到其他线程，当一个线程中的指令执行阻塞时，马上切换到其他线程。可以同时减少短延时和长延时造成的性能损失。但降低了单个线程的执行速度。

粗粒度多线程：仅在一个线程延时开销很大的情况下，才切换到另一个线程执行。

13 同时多线程：是多线程的一个特例，它使用多指令流出，动态调度处理器的资源开发线程级并行，同时开发指令级并行。

14 优先线程：允许多线程技术保持其优点的情况下，对单个线程的执行性能进行一些折中。

第七章 互连网络

1 互连网络：

互连网络是将集中式系统或分布式系统中的结点连接起来所构成的网络，这些结点可能是处理器、存储模块或者其它设备，它们通过互连网络进行信息交换。

静态网络：由点和点直接相连而成，这种连接方式在程序执行过程中不会改变。常用来实现一个系统中子系统或计算结点之间的固定连接。

动态网络：是用开关通道实现的，它可动态地改变结构，使其与用户程序中通信要求匹配。它能够根据程序的要求实现所需的通信模式。动态网络常用于集中式共享存储器多处理系统中。定时、开关和控制是动态互连网络的三个主要操作特征。

2 互连网络的性能参数：

结点度：与结点相连接的边的数目称为结点度。

网络直径： 网络中任意两个结点间最短路径长度的最大值称为网络直径。

等分宽度： 在将某一网络切成相等两半的各种切法中，沿切口的最小通道边数称为通道等分宽度

（对称网络：对于一个网络，如果从其中的任何一个结点看，拓扑结构都是一样的话，则称此网络为对称网络。）

维序： 按多维网络维序的特定顺序来选择后续通道。由于唯一性，可能产生死锁。

虚拟自适应： 将一个物理通道分成几个虚拟的通道，根据后续各虚拟通道的忙闲情况自适应选择后续通道。

路由（routing）： 在网络通信中对路径的选择与指定。常用的数据路由功能有：循环，置换，交换，均匀混洗，广播，选播，个人通信等。

虫孔路由：虫蚀（wormhole）把包进一步分成小片，硬件路由器有片缓冲区，同一个包中所有片象不可分离的同伴一样，以流水方式顺序传送。只有片头包含目标地址，所有片必须跟随片头。

3 静态连接网络的种类：

线性阵列，环和带弦环，循环移数网络，树形和星形，胖树形，网格形和环形网，超立方体，k 元 n-立方体网络。

4 动态连接网络：

阻塞网络：如果同时连接多个输入输出对时，可能会引起开关和通信链路使用上的冲突，这种多级网络称为阻塞网络。

非阻塞网络：如果多级网络通过重新安排连接方式可以建立所有可能的输入输出之间的连接，则称之为非阻塞网络。

1）总线：

实际上是一组导线和插座，用于处理与总线相连的处理器，存储器，和外围设备之间的数

据业务。总线系统造价较低，由于存在多个设备同时竞争总线，所以总线带宽较窄。

2）交叉开关网络：

在交叉开关网络中，每个输入端通过一个交叉点开关可以无阻塞地与一个空闲输出相连。对一个 n*n 的交叉开关网络来说，每一行可以同时接通多个交叉点开关，所以交叉开关网络的带宽和互连特性最好。但当 n 很大时，所需的硬件数量非常巨大。且交叉开关网络一旦构成后将不能扩充。

3）多级网络：

多级网络由一级以上的开关元件构成。这类网络可以把任一输入与任一输出相连。级间连接模式的选择取决于网络连接特性。多级网络的价格和性能介于总线系统和交叉开关望路之间。

多级网络可用于构造大型多处理机系统。每一级都用了多个 a×b 开关，相邻级开关之间都有固定的级间连接。为了在输入和输出之间建立所需的连接，可用动态设置开关的状态来实现。

动态连接网络中，总线，多级网络，交叉开关的特点的简单比较：

总线的造价最低，缺点是每台处理器可用的带宽较窄。另一个问题是容易产生故障。

由于交叉开关的硬件复杂性以 n^2 上升，所以其造价最为昂贵。且交叉开关网络一旦构成后将不能扩充。但是，带宽和路由性能最好。如果网络的规模较小，它是一种理想的选择。

多级网络则是两个极端之间的折衷。主要优点在于采用模块结构，因而可扩展性较好。然而其时延随网络的级数而上升。另外，由于增加了连线和开关复杂性，价格也是一种限制因素。

对多线程 MPP 系统进行分析考虑的主要参数及切换策略

考虑参数：远程通信时延

线程个数：线程用一个现场表示，现场由程序计数器，寄存器组和所要求的现场状态字构成。

现场切换开销

两次切换间隔

切换策略：Cache 失效时

每次加载时

执行每条指令时

执行指令快时。

《计算机体系结构》

名词解释总汇

集中式共享存储器结构(centralized shared-memory architecture)：这类多处理机在目前至多有几十个处理器。由于处理器数目较小，可通过大容量的 Cache 和总线互连使各处理器共享一个单独的集中式存储器。　　物理上分离的多个存储器可作为一个逻辑上共享的存储空间进行编址，这样一个处理器可以访问任何一个其他的局部存储器.这类机器的结构被称为分布式共享存储器(DSM，distributed shared-memory)或可缩放共享存储器（SSM，scalable shared-memory）体系结构。

整个地址空间由多个独立的地址空间构成，它们在逻辑上也是独立的，远程的处理器不能对其直接寻址。在这种机器的不同处理器中，相同的物理地址指向不同存储器的不同单元，每一个处理器-存储器模块实际上是一个单独的计算机，因而这种机器也称为多计算机(multicomputers)。

通讯延迟：发送开销+跨越时间+传输时间+接收时间

迁移是把共享数据项的拷贝放在一个局部的 Cache 中使用，从而降低了对远程共享数据的访问延迟。

复制是把多个处理器需要同时读取的共享数据项的拷贝放在各自局部 Cache 中使用，复制不仅降低了访存的延迟，也减少了对共享数据的访问产生的冲突。

目录（directory）——物理存储器中共享数据块的状态及相关信息均被保存在一个称为目录的地方。

监听(snooping)——每个 Cache 除了包含物理存储器中块的数据拷贝之外，也保存着各个块的共享状态信息。Cache 通常连在共享存储器的总线上，各个 Cache 控制器通过监听总线来判断它们是否有总线上请求的数据块。

在一个处理器写某个数据项之前保证它对此数据项有唯一的访问权，对应这种方法的协议称为写作废(**write invalidate**)协议。拥有唯一的 Cache 块拷贝的处理器通常称为这个 Cache 块的拥有者(owner)，处理器的写操作使自己成为对应 Cache 块的拥有者。

原子性(atomic)，即操作进行过程中不能被打断，例如将写失效的检测、申请总线和接收响应作为一个单独的原子操作。基于目录的相关性协议称为全映射(Full-map)。

原子交换（atomic exchange)：将一个存储单元的值和一个寄存器的值进行交换。建立一个锁，锁值为“0”表示开锁，为“1”表示上锁。

旋转锁是指处理器环绕一个锁不停地旋转而试图获得该锁。

栅栏（**barrier**）同步：是一个同步操作，它强制所有到达的进程进行等待，直到全部的进程到达栅栏，然后释放全部的进程，从而形成同步。

组合树是多个请求在局部结合起来形成树的一种分级结构，它降低冲突的原因是将大冲突化解成为并行的多个小冲突。可以排队记录等待的进程，当锁释放时送出一个已确定的等待进程，这种机制称为排队锁(queuing lock)。一个处理机对变量的写和另一个处理机对该变量的访问(读或写)由一对同步操作分开，其中一个在写操作后执行，另一个在别的处理机访问前执行，则称数据访问有序。

无同步操作排序变量可能提前被刷新，这种情况称为数据竞争(data race)，从而对于同步的程序可称之为无数据竞争(data-race-free)。

称与解锁相对应的同步操作为释放(release)，与加锁相对应的则称为获取(acquine)防护(fence)是计算过程中的固定点，用来保证无读或写穿过防护点。预取能返回最新数据值，并且保证对数据实际的存储器访问返回的是最新的数据项，则被称为非绑定的(nonbinding)

互连网络是将集中式系统或分布式系统中的结点连接起来所构成的网络，这些结点可能是处理器、存储模块或者其它设备，它们通过互连网络进行信息交换。静态网络由点和点直接相连而成，这种连接方式在程序执行过程中不会改变。

动态网络是用开关通道实现的，它可动态地改变结构，使其与用户程序中通信要求匹配。

与结点相连接的边的数目称为结点度(node degree)。链路或通道是指网络中连接两个结点并传送数字信号的通路。在单向通道的情况下，进入结点的通道数叫做入度(in degree)，而从结点出来的通道数则称为出度(out degree)，结点度是这两者之和。结点度应尽可能地小并保持恒定。网络中任意两个结点间最短路径长度的最大值称为网络直径。网络直径应当尽可能地小。

在将某一网络切成相等两半的各种切法中，沿切口的最小通道边数称为通道等分宽度 b (channel bisection width)。对于一个网络，如果从其中的任何一个结点看，拓扑结构都是一样的话，则称此网络为对称网络。

计算/通讯比：是衡量并行程序性能的尺度，是应用程序中相对于每次数据统需需要进行的计算。　　路由（routing）：在网络通信中对路径的选择与指定。置换（permutation）：指对象的重新排序。虫蚀（wormhole）：把包进一步分成小片，硬件路由器有片缓冲区，同一个包中所有片象不可分离的同伴一样，以流水方式顺序传送。只有片头包含目标地址，所有片必须跟随片头。

存储转发：是指每个结点有一个包缓冲区，包先进入缓冲区，当所需要的输出通道和接收结点的包缓冲区可用时，就将它传输给下一结点。

维序：按多维网络维序的特定顺序来选择后续通道。由于唯一性，可能产生死锁。

虚拟自适应：将一个物理通道分成几个虚拟的通道，根据后续各虚拟通道的忙闲情况自适应选择后续通道。

线性阵列(linear array)：是一种一维的线性网络，其中 N 个结点用 N-1 个链路连成一行。如果多级网络通过重新安排连接方式可以建立所有可能的输入输出之间的连接，则称之为非阻塞网络(nonblocking network)粗粒度：每台处理机所执行的程序为 20 秒以上，共享主存

中粒度：每台处理机所执行的程序为 10 毫秒以上，消息传递

细粒度：并行性高，在几个微秒量级，但通信开销大。

指令级并行（**Instruction - Level Parallelism(ILP)**）：指令序列中存在的潜在地并行性。

循环级并行性：循环体指令之间的并行性。

指令调度：通过改变指令在程序中的位置，将相关指令之间的距离加大到不小于指令执行延迟的时钟书，这样就可以将相关指令转化为无关指令。

循环展开：通过多次复制循环体并改变结束条件来相对增加有效操作时间。

名相关：如果两条指令使用相同的名，但是它们之间并没有数据流。包括反相关和输出相关。

反相关：指令 i 先执行，指令 j 写的名是指令 i 读的名。

输出相关：指令 j 和指令 i 写的名相同。

重命名技术：通过改变指令中操作数的名来消除名关。

控制相关：是指由分支指令引起的相关。

动态调度：通过硬件重新安排指令的执行顺序，来调整相关指令实际执行的关系，减少处理器的空转。

记分牌（**eboarding**）：是一项当有充足的资源和没有数据相关时允许指令乱序执行的技术。

寄存器重命名：一条指令流出时，存放操作数的寄存器被重命名为对应于该寄存器保留站的名称（编号）的过程。

动态分支预测：一种给予历史记录的分支预测，它解决记录一个分支指令的历史和决定预测的分支的一个问题的两个方面。

分支目标缓冲（**BTB**）：将分支成功的分支指令的地址和地分支目标地址都放到一个缓冲保存起来，缓冲区支指令的地址作为指示。

推断（Speculation）执行：允许在处理器还未判断指令是否能执行之前就提前执行，以克服控制相关。

保留站：用于保存等待流出和正在流出的指令所需的操作数。

在定序缓冲：加入指令确定极端的额外的硬件缓冲，保存指令执行完毕到指令得到确认之间的所有指令及结果。

超标量（Superscalar）：每个时钟流出的指令输不定。

超流水（Super Pipeling）：是将每个功能部件进一步流水化，是的一个功能部件在一拍中可以处理多条指令。

超长指令字 VLIW（Very Long Instruction Word）：每个时钟周期流出的指令数是固定的，它们构成一条长指令，或者是一个混合的指令包。

DLX 标量：每个时钟流出两条指令。

The compiler technique to create additional instruction-level parallelism for a loop is simply called, ***loop unrolling.***

The hardware technique to create additional instruction-level parallelism for a loop is simply called, ***register renaming.***

Reservation stations: buffers hold instructions and operands that have been issued and are awaiting execution at a functional unit

A recurrence is when a variable is defined based on the value of that variable in an earlier iteration, often the one immediately preceding, as in the above fragment.

As an example, a simple and sufficient test for the absence of a dependence is **the greatest common divisor(GCD) test.**

软件流水：是一项重构造相互重叠进行地软件流水线代码地循环，使其指令从原始的循环中的不同重复中选取的技术。

路径调度是用一项通过不同于循环分支的条件分支发觉并行的技术，扩展科循环展开。

路径：试图去发觉一个可能的其操作将被放入一个小数目的指令集基本程序块的顺序称为路径（Trace）。选择此路径称为**路径选择（trace selection）**。

路径精简：试图去精简路径到一个小数目的广泛的指令集的过程。（**trace compaction**）

A set of status bits, called **poison bits**, are attached to the result registers written by speculated instructions when the instructions cause exceptions.

An alternative is to move instructions past branches, flagging them as speculative, and providing renaming and buffering in the hardware, much as Tomasulo's algorithm does. This concept has been called **boosting(推进)**

Adding this commit phase to the instruction execution sequence requires some changes to the sequence as well as an additional hardware buffer, called the reorder buffer, to hold the results of instructions that have finished execution but have not committed.

缓冲中保存起来，缓冲区分支指令的地址作为标示。

前瞻（speculation）执行：允许在处理器还未判断指令是否能执行之前就提前执行，以克服控制相关。

保留站：用于保存等待流出和正在流出的指令所需的操作数。

再定序缓冲：在前瞻执行的指令之间传送结果的一套额外的硬件缓冲，保存指令执行完毕到指令得到确认之间的所有指令及结果。

超标量（superscalar）：每个时钟流出的指令不定。

超流水（super pipeling）：是指每个功能部件进一步流水化，使得一个功能部件在一拍中可以处理多条指令。

超长指令字 VLIW（very long instruction word）：每个时钟周期流出的指令数是固定的，它们构成一条长指令，或者是一个混合的指令包。

DLX 标量：每个时钟流出两条指令。

The compiler technique to create sdditional instruction-level parallelism for a loop is simply called loop unrolling.

The hardware technique to create additional instruction-level parallism for a loop is simply called register renaming.

Reservation stations: buffers hold instructions and operands that have been issued and are awaiting execution at a functional unit.

A recurrence is when a variable is defined based on the value of that variable in an earlier iteration, often the one immediately preceeding, as in the above fragment.

As an example, a simple and sufficient test for the absence of a dependence is the greatest common divisor (GCD) test.

软件流水：是一项重构造相互重叠进行的软件流水性代码的循环，使其指令从原始的循环中的不同重复中选取的技术。

路径调度是用一项通过不同于循环分支的条件分支发觉并行的技术，扩展可循环展开。

路径：试图去发觉一个可能的其操作将被放入一个小数目的指令集基本程序块的顺序称为路径（trace），须子此路径称为路径选择（trace selection）。

路径精简：试图去精简路径到一个小数目的广泛的指令集的过程（trace compaction）

a set of status, called poison bits, are attached to the result registers written by speculated instructions when the instructions cause exceptions.

An alternative is to move instructions past branches, flagging them as speculative, and providing renaming and buffering in the hardware, much as Tomasulo's algorithm does. This concept has been called boosting（推进）.

Adding this commit phase to the instruction execution sequence requires some changes to the sequence as well as an additional hardware buffer, called the reorder buffer, to hold the results of instructions that have finished execution but have not committed.

3．1 术语

1. 流水线：将一个重复的时序过程，分解为若干个子过程，而每一个子过程都可有效地在其专用功能段上与其他子过程同时执行。
2. 单功能流水线：只能完成一种固定功能的流水线。

3. 多功能流水线：流水线的各段可以进行不同的连接，从而使流水线在不同的时间，或者在同一时间完成不同的功能。
4. 静态流水线：同一时间内，流水线的各段只能按同一种功能的连接方式工作。
5. 动态流水线：同一时间内，当某些段正在实现某种运算时，另一些段却在实现另一种运算。
6. 部件级流水线：（运算操作流水线）把处理机的算术逻辑部件分段，以便为各种数据类型进行流水操作。
7. 处理机型流水线：（指令流水线）把解释指令的过程按照流水方式处理。
8. 处理机间流水线：（宏流水线）由两个以上的处理机串行地对同一数据流进行处理，每一个处理机完成一项任务。
9. 线形流水线：指流水线的各段串行连接，没有反馈回路。
10. 非线形流水线：指流水线中除有串行连接的通路外，还有反馈回路。
11. 标量流水处理机：处理机不具有向量数据表示，仅对标量数据进行流水处理。
12. 向量流水处理机：处理机具有向量数据表示，并通过向量指令对向量的各元素进行处理。
13. 结构相关：某些指令组合在流水线中重叠执行时，长生资源冲突，则称该流水线有结构相关。
14. 数据相关：当指令在流水线中重叠执行时，流水线有可能改变指令读/写操作的顺序，使得读/写操作顺序不同于它们非流水实现时的顺序，将导致数据相关。
15. 定向：将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，或所有需要它的功能单元，避免暂停。

两条指令 i,j，i 在 j 前进入流水线。

16. RAW:j 执行要用到 i 的结果，但当其在流水线中重叠执行时，j 可能在 I 写入其结果之前就先行对保存该结果的寄存器进行读操作，得到错误值。
17. WAW：j、I 的操作数一样，在流水线中重叠执行时，j 可能在 I 写入其结果之前就先行对保存该结果的寄存器进行写操作，导致写错误。
18. WAR：j 可能在 I 读某个寄存器之前对该寄存器进行写操作，导致 I 读出数据错误。

3．2 答：1. 流水过程由多个相联系的子过程组成。

2. 每个子过程由专用的功能段实现。
3. 各个功能段所需时间尽量相等。
4. 流水线有“通过时间”（第一个任务流出结果所需的时间）。在此之后流水过程才进入稳定工作状态，一拍流出一个结果。
5. 流水技术适合于大量重复的时序过程，只有输入端连续提供任务，流水线效率才可充分发挥。

3．3 答：工作原理：把一条 DLX 指令在 5 个周期内实现，将每一个时钟周期看作是流水线的一个时钟周期，硬件每个时钟周期启动一条新的指令，并执行 5 条不同指令中的某一部分。每条指令虽仍需 5 个时钟周期完成，但提高了吞吐率，实现了流水。

instr./clock	1	2	3	4	5	6	7	8	9

I	IF	ID	EX	MEM	WB			
I+1		IF	ID	EX	MEM	WB		
I+2			IF	ID	EX	MEM	WB	
I+3				IF	ID	EX	MEM	WB
I+4					IF	ID	EX	MEM	WB

3．4 答：指令多周期实现可以降低时钟周期时间，单周期实现则可降低 CPI，同时延长时钟周期。单周期实现可以省去一些临时寄存器，但是对多数机器而言，单周期实现并非十分有效，因为不同指令完成的操作与所需时钟周期时间都不同。单周期实现必须重复设置指令执行功能部件，多周期实现可采用流水技术共享功能单元。

3．5 答：（1）流水化功能单元

（2）资源重复

3．6 解：（1）在流水线中尽早判断出分支转移是否成功；

（2）尽早计算出分支转移成功时的 PC 值（即分支的目标地址）

- “冻结”“排空”流水线的方法
- 预测分支失败
- 预测分支成功
- 延迟分支

3．7 答：1、从前调动：分支必须不倚赖于被调度的指令，总是可以有效提高流水线性能。

2、从目标处调度：若分支转移失败，必须保证被调度的指令对程序的执行没有影响，可能需要复制被调度指令。分支转移成功时，可提高流水线性能。但由于复制指令，可能加大程序空间。

3、从失败处调度：若分支转移成功，必须保证被调度的指令对程序的执行无影响。分支转移失败时，可提高流水线性能。

3．8 答：1、水平处理方式：若向量长度为 N，则水平处理方式相当于执行 N 次循环。若使用流水线，在每次循环中可能出现数据相关和功能转换，不适合对向量进行流水处理。

2、垂直处理方式：将整个向量按相同的运算处理完毕之后，再去执行其他运算。适合对向量进行流水处理，向量运算指令的源/目向量都放在存储器内，使得流水线运算部件的输入、输出端直接与存储器相联，构成 M-M 型的运算流水

线。

3、分组处理方式：把长度为 N 的向量分为若干组，每组长度为 n，组内按纵向方式处理，依次处理各组，组数为 $\left\lceil \frac{N}{n} \right\rceil$，适合流水处理。可设长度为 n 的向量寄存器，使每组向量运算的源/目向量都在向量寄存器中，流水线的运算部件输入、输出端与向量寄存器相联，构成 R-R 型运算流水线。

指令级并行：指令序列中存在着潜在并行性称为指令级并行。
指令调度：通过改变指令在程序中的位置，将相关指令之间的距离加大到不小于指令执行延迟的时钟数，即将相关指令转化为实际无关的指令。
循环展开：通过展开循环体若干次，将循环级并行性转化为指令级并行
名相关：两条指令使用相同的名，但他们之间没有数据流，称之为名相关。
A、输出相关：两条指令写相同的名，即写后写相关。
B、反相关：指令1先执行，指令j写的名是I读的名，即读后写相关。
动态调度：通过硬件重新安排指令的执行循序，来调整相关指令实际执行时的关系，减少处理机空转。
静态调度：通过编译器调度导致阻塞相关的指令，减少处理机空转。
记分牌：在指令动态调度中，通过相应电路检测到指令运行所需的资源满足并没有数据相关，就允许指令乱序执行同时记录下这些指令的运行状态。
寄存器重命名：使用大量的虚拟寄存器来替代源代码中的寄存器。
Tomasulo 算法：通过动态调度、寄存器重命名、动态存储器地址判别技术来解决 waw 相关和 war 相关。
保留站：保存等待流出和正在流出指令所需的操作数。
公共数据总线：计算结果通过相关专用通路直接从功能部件到对应的保留站进行缓冲，相关专用通路通过一条公用数据总线来实现。
分支预测缓冲(BPB)：使用一片缓冲区记录最近一次或几次的分支历史，与分支指令地址的低位来索引，缓冲的存储区为1位的分支历史记录位，称为预测位。
分支目标缓冲（BTB）：将分支成功的分支指令和它的分支目标地址都放到一个缓冲区内保存起来，缓冲区以分支指令的地址作为标识。
推断执行：处理机还未判断指令是否能执行之前就提前执行，即克服了控制相关。
再定序缓冲：在推断执行中为了实现乱序执行但顺序确认，故加入指令确认阶段需要一套额外的硬件缓冲，来保存那些执行完但未经过确认的指令及其结果，这个硬件的缓冲称为再定序缓冲，同时还用他来进推断执行的指令之间传送结果。
超标量（superscalar）：即每个时钟周期流出的指令数是不定，它既可以通过编译器静态调度，也可以通过记分牌或 tomasulo 算法等动态调度实现多指令流出。
超流水（superpileling）：是将每个功能部件进一步流水化，使得一个功能部件在一拍中可以处理多条指令。
超长指令字（VLIW）：每个时钟周期流出的指令数是固定的，他们构成一条长指令，或者是一个预先的指令包。
Trace compaction:Once a trace is selected is selected ,the second process called trace compaction. It attempts to move operations as early as it can in a sequence(trace), packing the operations into a few wide instructions as possible.
集中式共享存储器（CSMA）：处理器数目较小，可通过大容量的 cache 和总线使各处理器共享一个单独的集中式存储器。
分布式共享存储器(DSM，distributed shared-memory)或可缩放共享存储器（SSM，scalable shared-memory）体系结构：物理上分离的多个存储器可作为一个逻辑上共享的存储空间进行编址，这样一个处理器可以访问任何一个其他的局部存储器。
通讯延迟：发送开销+跨越时间+传输时间+接收时间
计算/通讯比：是衡量并行程序性能的尺度。
超结点：每个结点内还可能包含较小数目(2-8)的处理器，这些处理器之间可采用另一种技术（例如通过总线）互连形成簇(cluster)，这样形成的结点叫做超结点。
迁移：是把远程共享数据项的拷贝放在本处理器的一个局部 Cache 中使用，从而降低了对远程共享数据的访问延迟。
复制：是把多个处理器需要同时读取的共享数据项的拷贝放在各自局部 Cache 中使用，这不仅降低了访存的延迟，也减少了对共享数据的访问产生的冲突。
目录（directory）：物理存储器中共享数据块的状态及相关信息均被保存在一个称为目录的地方。
监听(snooping)：每个 Cache 除了包含物理存储器中块的数据拷贝之外，也保存着各个块的共享状态信息。Cache 通常连在共享存储器的总线上，各个 Cache 控制器通过监听总线来判断它们是否有总线上请求的数据块。
写作废（write invalidate）协议：在一个处理器写某个数据项之前保证它对此数据项有唯一的访问权。
写更新协议：当一个处理器写某数据项时，通过广播使其它 cache 中所有对应的该数据项拷贝进行更新，原子性(atomic)，即操作进行过程中不能被打断。
互联网络：是将集中式系统或分布式系统中的节点连接起来所构成的网络。
结点度：与结点相连接的边的数目称为结点的度。
网络直径：网络中任意两个结点间最短路径长度的最大值称为网络直径。虚拟自适应：将一个物理通道分成几个虚拟的通道，根据后续各虚拟通道的忙闲情况自适应选择后续通道。拥有者（owner）：拥有唯一的 Cache 块拷贝的处理器通常称为这个 Cache 块的拥有者(owner)，处理器的写操作使自己成为对应 Cache 块的拥有者。
等分带宽：去将某一网络切成相等两半的各种切法中，沿切口的最小通道边数称为通道等分宽度。
路由（routing）：在网络通信中对路径的选择与指定。
静态网络：由点和点直接相连而成，这种连接方式在程序执行过程中不会改变。
动态网络：是用开关通道实现的，它可动态地改变结构，使其与当前程序中通信要求匹配。
栅栏（barrier）同步：是一个同步操作，它强制所有到达的进程进行等待，直到全部的进程到达栅栏，然后释放全部的进程，从而形成同步。
旋转锁：指处理机环绕一个锁不停地旋转而请求获得该锁。

虫孔路由：虫蚀（wormhole）把包进一步分成小片，硬件路由器有片缓冲区，同一个包中所有片象不可分离的同伴一样，以流水方式顺序传送。只有片头包含目标地址，所有片必须跟随片头。

1、对多线程 MPP 系统进行分析考虑的主要参数以及多线程的切换策略。

答：时延，即远程通信时延；线程个数，线程用一个现场表示，现场由程序计数器、寄存器组和所要求的现场状态字构成；现场切换开销；两次切换的间隔。

切换策略：在 Cache 失效时；每次加载时；执行每条指令时（相关性小，并行好）；执行指令块时；

2、简单比较动态网络中总线、多级网络、交叉开关的特点。

答：构成动态网络的总线、多级网络、交叉开关中，总线的造价最低，但其缺点是每台处理器可用的带宽较窄。总线所存在的另一个问题是容易产生故障。有些容错系统，如用于事务处理的 Tandem 多处理机等，常采用双总线以防止系统产生简单的故障。

由于交叉开关的硬件复杂性以 N^2 上升，所以其造价最为昂贵。但是，交叉开关的带宽和路由性能最好，如果网络的规模较小，它是一种理想的选择。

多级网络则是两个极端之间的折衷，它的主要优点在于采用模块结构，因而可扩展性较好，然而，其时延随网络的级数而上升。另外，由于增加了连线和开关复杂性，价格也是一种限制因素。

3、什么是多处理机的相关性（coherency）和一致性（consistency）？给出解决相关性的监听协议的工作原理。

答：如果对某个数据项的任何读操作均可得到其最新写入的值，则认为这个存储系统是一致的。包括了两个不同方面：相关性：返回给读操作的是什么值(what, Coherence)；一致性：什么时候才能将已写入的值返回给读操作(when, Consistence)。

4、给出四种松弛（relaxed）一致性模型的特点及其实现上所需要的硬件支持措施。

答：第一个模型是松弛写和读(对不同的地址)的顺序，即取消 W—R 顺序，这种模型采用写缓存，并提供读的旁路机制，从而允许处理机在其写的操作被所有的别的处理机看到之前就继续运行读；如果松弛 W—W，允许非冲突写隐含地乱序进行，则成为部分存序(partial store ording PSO)模型，从实现的角度看，可以使写流水化或重叠，而不是强制一个操作必须在另一个之前结束，对同步操作仍需将写操作挂起，因为它引起等防护；松弛的模型的第三类是取消 R—R，R—W，W—R 和 W—W，这种模型称为弱排序(Weak ordering)；进一步弱化的模型称为释放一致性(release consistency)模型，这种模型区分同步操作中的访问一个共享变量的获取操作 SA 和将对象释放允许别的处理机获取访问权的释放操作 SR。

Tomasulo算法和Scoreboard区别：<两种何为动态调度算法>

1. Tomasulo算法采用了寄存器重命名技术
2. Tomasulo算法冲突检测和指令执行控制机制分开。
   一个功能部件的指令何时开始执行，由每个功能部件的保留站控制
   而记分牌是集中控制
3. Tomasulo算法中，计算结果通过 CDB 送往保留站缓冲，记分牌将结果写入寄存器，等待功能部件来相互竞争。

Scoreboard 技术要点：顺序流出，乱序执行

1. IS阶段检查结构相关（功能部件空闲）和 WAW 相关
2. RO阶段检查 RAW 相关
   · EXE阶段完毕：
3. WR阶段检查 WAR 相关。

记分牌中记录所有的寄存器（源）全部使用如何以读操作数。

Qj，Qk：若源没准备好，则源是由哪个部件产生。

Rj，Rk：源寄存器已准备好（如：Yes 没有：No）。或已取走（No）

Fi，Fj，Fk：（目的，源1，源2）寄存器

Tomasulo算法：允许指令发生冲突后还可以继续执行

1. IS：检查结构冲突，进行寄存器重命名处理
2. EXE：检查先写后读数据冲突，两个操作数都有效时执行。
3. WR：写入CDB。

Qj，Qk：产生结果的保留站号

Vj，Vk：源操作数的值。

Qi：操作结果要存入本寄存器（存储器）的指令的保留站号

它提高指令很并行的关键技术：动态调度，寄存器重命名，动态存储器地址判别。

控制相关的动态解决技术：

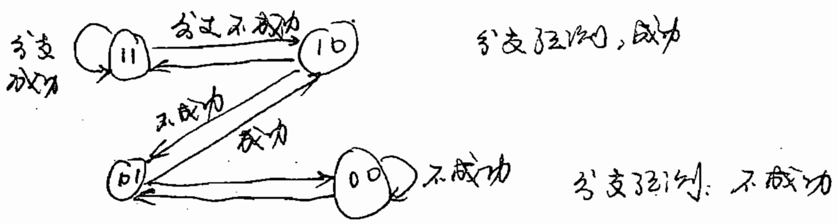
一、减少分支延迟：分支预测缓冲技术

1. 最简单的动态分支预测技术 BPB。只有在~~分支成功时~~ 转移延迟大于传送PC时间才能减少开销。

0、N位分支预测：计数器的值在 $0 \sim 2^N-1$ 之间变动

当计数器的值 $\geq (2^N-1)/2$：预测成功，否则预测为不成功。

实际使用中，N位预测和2位预测的效果相同。



两位分支预测的准确率：使用4096个记录项的BPB下，与无穷大的Buffer性能相当。

二、BTB：分支目标缓冲处理步骤和分支延迟的计算。

三、基于硬件的推断执行：乱序执行，顺序确认。

结合了三种思想：① 动态的分支预测，执行哪条语句

② 在控制相关消除之前指令推断执行

③ 基本块采用动态调度

reorder buffer：保存指令执行完毕到指令得到确认之间所有指令及其结果。如Tomasulo算法中的 reservation station 一样是后续指令操作数的来源之一。集成了 Load，store 缓冲的功能。

1) -IS阶段：保留站和再定序缓冲都空

2）执行：两个操作数全有效才执行。RAW相关检测

3）写结果：结果有效后写到 CDB，再从 CDB 写到 reorder buffer 或 reservation station。其中，reservation station 既可以从 reorder buffer 读数，又可从 CDB 直接读数。

4）commit：如果分支指令预测正确，到达 reorder buffer 出口且结果有效时，将结果写回寄存器，再 clear reorder buffer；预测错误，当指令到达出口时，clear reorder buffer，从正确处开始。

四：多指令流出技术：

1：多指令流出处理器有3种：

超标量，超流水，超长指令字（superscale，superpipeline，VLIW）

超标量：每个时钟周期流出的指令数不定，既可编译器静态调度又可通过 scoreboard 和 Tomasulo 动态调度。（动态多流出）

超长指令字：每个时钟周期流出的指令数是固定的，构成一条长指令。目前只能通过 compiler 静态调度。（静态多流出）

2. 多指令流出的动态调度：

硬件调度机制能流出相关的两条指令。

五. 大规模机器同步有哪些软件或硬件支持方法？ 答：无竞争条件下延迟较小，激烈竞争时串行性小。主要是软件实现来提高锁和栅栏的性能。

答 软件实现：① 旋转锁实现，当进程加锁失败时，就人为推迟这些进程的等待时间（指数）

② 排队锁实现。数组实现（用组合树结构，多个请求在局部结合起来形成树的一种分级结构）

硬件支持：采用硬件同步原语，一种针对锁，另一种针对栅栏和要求进行计数或提供明确索引的某些用户操作。

目录协议的实现：

1. 目录：用一种专用的存储器所记录的数据结构，它记录着可以进入 Cache 的每个数据块的访问状态，该块在各个处理器的共享状态，以及是否修改过等信息。

分布式目录：目录一般与存储器一起分布在系统中。

2. 数据块的状态：① 共享：一个或多个 processor 有，存储器中数据最新

② 未缓存：无任何 processor 有它的副本

③ 独占：只有一个 processor 有它的拷贝，存储器中副本无效。

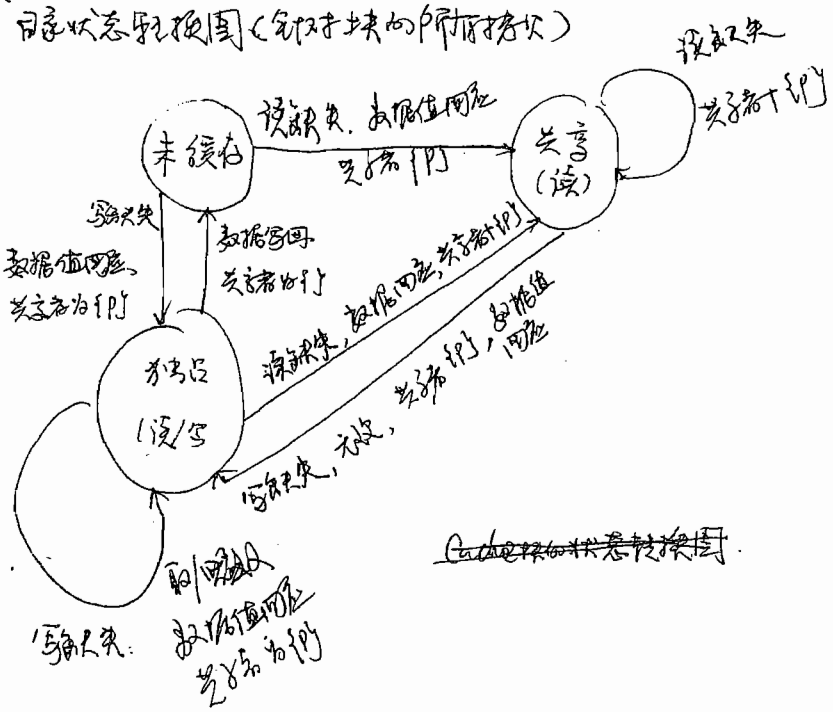
本地节点：产生请求的节点。

宿主(home)节点：存储地址和目录条目所在的节点。

远程节点：拥有高速缓存块副本的节点。

基本原理：

1. 目录状态转换图（针对块的所有拷贝）



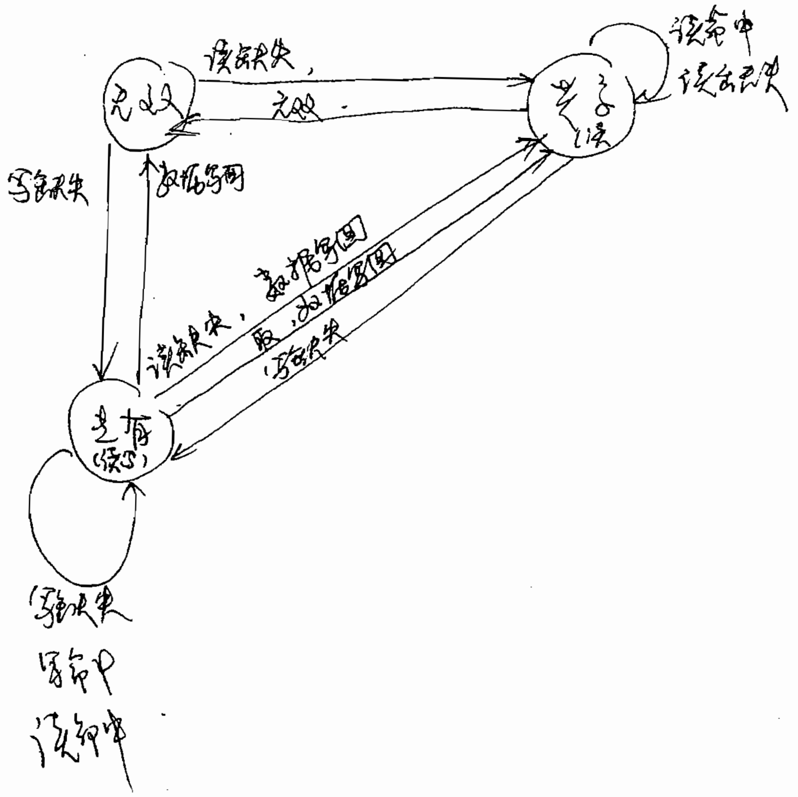
~~Cache 块的状态转换图~~

2. 为每个结点增加目录存储器用于存放目录项

目录项的组成：① 状态：描述该目录对应存储块当前状态

② 位向量：有 N 位，每位指一个处理器的局部 cache，指示 Cache 中有无该块的拷贝。

3. 目录协议中数据块的状态转换



监听协议：

某个处理器写数据时，先获得总线，将作废数据块地址放到总线，其它节点监听总线，检测地址是否在它们的Cache中，在，作废。获取总线的顺序保证了写的顺序。

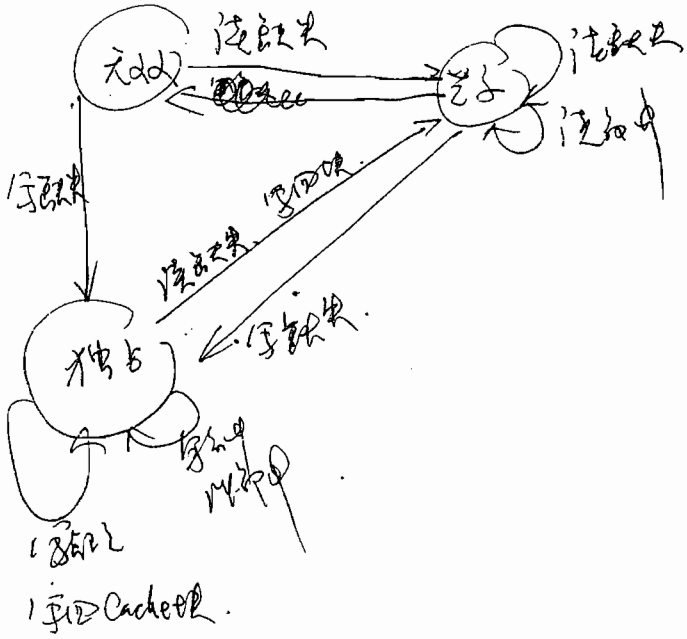
Cache未命中，作废其它Cache块，写直达时从Memory中取。

写回：(设置共享位，修改位)。在回写各个processor总线，监听的数据块的processor返回数据，修改其状态

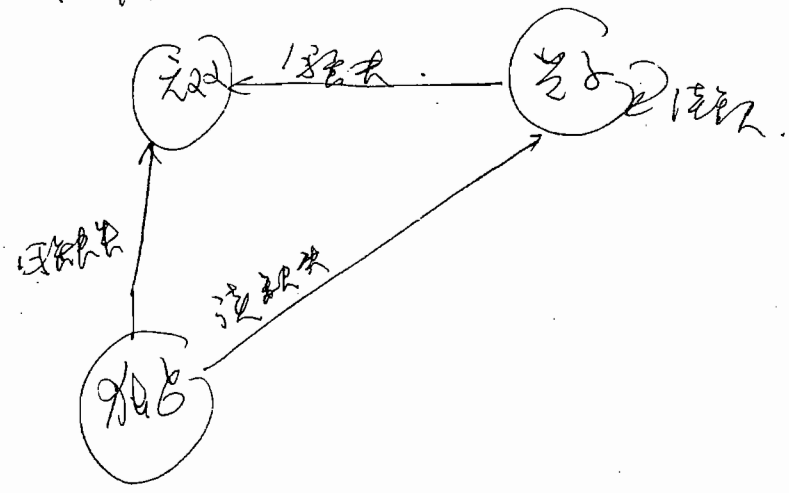
{ 有共享，地址放总线
无共享，不放

修改了位置1 → 作废

等于cpu转成和Cache交换数据.



等于总线和单个Cache的状态.



附加题.