计算机系统结构复习平板版

面向 iPad 阅读和批注：整理版正文 + 公式 + 原题练习卡。网页版本仍保留在同一文件夹中。

1. 总览串讲

把整门课压成一条复习主线：先明确系统结构研究什么，再用定量方法评价性能，最后逐个拆解处理器、存储、I/O、互联网络和多处理器。

1. 一条主线

计算机系统结构这门课可以统一理解成一个问题：如何设计一个计算系统，让它在功能、性能、成本、功耗、可靠性和可编程性之间取得合理平衡。所有章节都围绕这个目标展开。

先定义接口

ISA 规定软件能看到什么：指令、寄存器、寻址方式、异常和中断等。它是软件和硬件之间的契约。

再提高执行效率

流水线、分支预测、多指令流出等技术都在开发指令级并行，目标是降低 CPI、提高吞吐率。

最后处理系统瓶颈

存储墙、I/O 可靠性、互联网络带宽、多处理器同步，都是系统层面的性能和正确性问题。

2. 三个层次：结构、组成、实现

层次	关心的问题	例子
计算机系统结构	机器对程序员和编译器呈现什么功能，属于“做什么”。	是否有 ADD 指令，指令格式、寄存器数量、寻址方式。
计算机组成	这些功能用什么逻辑部件和控制流程实现，属于“怎么做”。	ALU、寄存器堆、控制信号、数据通路。
计算机实现	组成方案如何落到具体电路、工艺、物理结构上，属于“做出来”。	CMOS 门电路、版图、32 位加法器实现。

考试如果要求解释三者关系，抓住“系统结构是软件可见接口，组成是逻辑实现，实现是物理实现”就不容易跑偏。

3. 定量分析是全课工具箱

系统结构设计不能只凭直觉，需要用执行时间、吞吐率、CPI、失效率、可靠度等指标做比较。后面每一章的计算题，本质上都是把具体问题转化成这些指标。

以经常性事件为重点

优化高频路径最划算。哪一部分占总时间比例越大，优化它对总性能的影响越明显。

Amdahl 定律

加速比受可改进部分比例限制。可并行部分少时，处理器数量再多也不会得到理想加速。

CPU 性能公式

从 IC、CPI、时钟周期时间三个量拆解 CPU 时间，方便定位优化来自哪里。

局部性原理

时间局部性和空间局部性是 Cache、预取、存储层次能有效工作的基础。

CPU 时间 = IC × CPI × 时钟周期时间

Amdahl 加速比 = 1 / ((1 - f) + f / s)

4. 建议复习路线

5. 全课公式与题型总表

章节	必会公式	典型题型
基本概念	CPU 时间 = IC x CPI x 时钟周期；Speedup = 1 / ((1 - f) + f / s)	Amdahl 反求比例；平均 CPI；MIPS/MFLOPS。
流水线	Tk = (k + n - 1) Δ t；TP = n/Tk；S = T顺序/T流水；E = 有效时空面积/总时空面积	等时/不等时流水线；时空图；非线性调度。
指令级并行	CPI流水线 = CPI理想 + 各类停顿；理想 CPI 约为 1/流出宽度	BHT 状态更新；BTB 延迟；超标量/VLIW/超流水线比较。
存储系统	AMAT = Hit Time + Miss Rate x Miss Penalty；两级 AMAT = H1 + MR1(H2 + MR2P2)	地址字段；Cache 策略；两级 Cache；主存带宽；TLB/脏块写回。
I/O 外存	串联 R = ΠRi；并联 R = 1 - Π(1 - Ri)；Availability = MTTF/(MTTF+MTTR)	可靠度；RAID 小写；RAID10/01 对比。
互联网络	超立方体距离 = 汉明距离；Omega 级数 = log2N；开关数 = (N/2)log2N	互连函数变换；最短路；Omega 寻径。
多处理器	并行加速比 = 1 / ((1 - f) + f / p)；CPI = 本地 CPI + 通信/同步停顿	并行上限；远程访问；MSI 状态；同步开销。

自测问题

合上正文后先试着口答，再展开看答案。能把这些问题讲清楚，本章主线基本就稳了。

系统结构回答“对软件承诺什么功能”，组成回答“用什么逻辑部件实现”，实现回答“落到什么物理电路/工艺”。

因为系统设计需要比较执行时间、CPI、吞吐率、失效率、可靠度等指标，避免只凭直觉判断优化是否值得。

局部优化收益受该部分原始时间占比限制，优化低频部分通常不会显著提高整体性能。

时间局部性和空间局部性让热点数据更可能在上层存储命中，因此 Cache/主存/辅存层次才有效。

先看总览主线，再集中公式计算，最后用各专题的概念对照和自测题查漏补缺。

2. 基本概念

这一页负责打地基：性能是什么意思，怎么计算加速比，CPU 时间、CPI、MIPS、MFLOPS 各自怎么用，以及为什么基准测试要谨慎比较。

1. 本章定位

基本概念章是后续所有计算题的入口。流水线、Cache、多处理器看起来题型不同，但最终常常会回到执行时间、CPI、吞吐率、加速比这些量。

响应时间

单个任务从开始到完成的时间，个人用户最关心。越短表示性能越好。

吞吐率

单位时间完成的任务数量，服务器和批处理系统更关心。越大表示系统处理能力越强。

加速比

改进前执行时间除以改进后执行时间，用来衡量优化带来的整体收益。

2. Flynn 分类

类型	含义	复习要求
SISD	单指令流、单数据流，传统单处理器模型。	了解。
SIMD	单指令流、多数据流，同一操作作用于多个数据。	能和向量处理、GPU 直觉联系起来。
MISD	多指令流、单数据流，现实中少见。	了解即可。
MIMD	多指令流、多数据流，现代多处理器和集群常见。	后面多处理器章节会用到。

3. Amdahl 定律

Amdahl 定律的核心直觉是：优化收益不仅取决于局部优化有多快，还取决于这个局部在总执行时间中占多大比例。

$$\text{Speedup} = 1 / ((1 - f) + f / s)$$

符号	含义	做题提醒
f	可改进部分原来占总执行时间的比例。	题目说“某部件占总时间 40%”，通常就是 $f = 0.4$。
s	可改进部分自身被加速的倍数。	题目说“速度提高到原来的 5 倍”，通常就是 $s = 5$。
$1 - f$	不可改进部分。	这部分决定加速比上限，千万不能丢。

4. CPU 时间、CPI 与平均 CPI

CPU 时间 = IC × CPI × 时钟周期时间 = IC × CPI / 时钟频率

这个公式最适合分析处理器或存储优化对程序执行时间的影响。后面 Cache 题经常把存储器停顿周期加到 CPI 里。

指标	含义	常见计算
IC	Instruction Count，执行的指令条数。	和算法、编译器、ISA 有关。
CPI	Cycles Per Instruction，每条指令平均时钟周期数。	不同类型指令 CPI 不同，用加权平均。
时钟周期时间	一个时钟周期的长度。	等于 1 / 时钟频率。

平均 CPI = Σ (某类指令比例 × 该类指令 CPI)

5. MIPS 与 MFLOPS

指标	公式	适用范围
MIPS	指令条数 / (执行时间 × 10^6)	同一程序、同类机器下可参考；跨 ISA 容易误导。
MFLOPS	浮点操作次数 / (执行时间 × 10^6)	更适合浮点密集型程序；不同程序浮点操作复杂度也可能不同。

如果题目要求比较真实性能，优先使用执行时间；如果只是给定指令数和时间求指标，再套 MIPS/MFLOPS。

6. 性能评价原则

- 评价计算机性能时，应尽量使用真实程序或代表性基准程序。
- 同一系统对不同应用的表现可能差别很大，不能只看单一指标。
- 比较优化方案时，要明确比较的是响应时间、吞吐率、能耗还是成本。
- 如果多个测试程序合成一个结果，注意算术平均、几何平均的含义差异。

基准程序可以分为真实应用程序、核心程序、小型基准程序和合成基准程序。越接近真实负载，评价越可信；越小越容易分析，但也越可能偏离真实表现。

7. 未来趋势与墙

资料中还提到体系结构发展的若干“墙”和机遇。它们不是本章计算重点，但能帮助理解为什么后续章节要研究存储层次、功耗、互联和并行。

趋势/问题	含义	关联章节
存储墙	处理器速度提升快于存储器访问速度，访存成为瓶颈。	Cache、主存、虚拟存储。
功耗墙	频率继续提升会带来功耗和散热压力。	多核、多处理器、能效设计。
带宽墙	数据搬运能力限制系统吞吐。	主存带宽、I/O、互联网络。

趋势/问题	含义	关联章节
应用墙	硬件能力增长需要应用并行性和软件生态配合。	指令级并行、多处理器。
硬件设计大众化与云计算	工具链和云资源降低硬件创新门槛。	体系结构未来机会。

8. 公式与习题精讲

Amdahl 多部件改进通式

$$S = 1 / ((1 - \Sigma fi) + \Sigma (fi / si))$$

fi 是第 i 个可改进部件在原总执行时间中的比例，si 是该部件自身加速比。若题目反求某个比例，把目标总加速比代入即可。

例题模板：反求可改进比例

部件1、2、3加速比分别为 30、20、10。若 f1 = 0.3，f2 = 0.3，要求总加速比 S = 10，求 f3。

$$0.1 = (1 - 0.3 - 0.3 - f3) + 0.3/30 + 0.3/20 + f3/10$$

$$0.1 = 0.425 - 0.9f3，所以 f3 = 0.325 / 0.9 = 65/180 \approx 0.36$$

例题模板：改进后不可加速部分占比

若 f1 = 0.3，f2 = 0.3，f3 = 0.2，三部分分别加速 30、20、10 倍，则改进后总时间为：

$$Tnew = 0.3T/30 + 0.3T/20 + 0.2T/10 + 0.2T = 0.245T$$

$$不可加速部分在改进后时间中占比 = 0.2T / 0.245T \approx 0.82$$

平均 CPI 例题套路

$$CPIavg = \Sigma (CPIi \times ICi/IC)$$

注意：CPI 公式中的比例是“指令条数比例”，而 Amdahl 里 fi 是“时间比例”。两者不能混用。

例题模板：比较两种优化方案

FP 指令占 25%，非 FP 占 75%；FPSQR 占全部指令 2%，它属于 FP。改进前 CPI_FP = 4.0，CPI_非FP = 1.33，CPI_FPSQR = 20。

先由改进前平均 CPI 反求普通 FP 指令 CPI：

$$4 \times 0.25 + 1.33 \times 0.75 = 20 \times 0.02 + CPI\_非FPSQR \times 0.98$$

$$CPI\_非FPSQR = 80/49$$

方案一：只把 FPSQR 的 CPI 从 20 降到 2。

$$CPI1 = 2 \times 0.02 + (80/49) \times 0.98 = 1.64$$

方案二：把所有 FP 指令 CPI 从 4 降到 2。

$$CPI2 = 2 \times 0.25 + 1.33 \times 0.75 \approx 1.50$$

所以方案二更好。

MIPS/MFLOPS 计算

$$MIPS = 指令条数 / (执行时间 \times 10^6) = 时钟频率 / (CPI \times 10^6)$$

$$MFLOPS = 浮点操作次数 / (执行时间 \times 10^6)$$

如果题目问“哪个机器更快”，优先比较执行时间；如果题目明确给指令数/浮点操作数和时间，再计算 MIPS/MFLOPS。

9. 原题练习区

【原题】Amdahl 定律：三个部件同时改进

原题 1.4.1；原题截图：p.6

计算机系统有三个部件可以改进，三个部件的加速比分别为：部件加速比1 = 30，部件加速比2 = 20，部件加速比3 = 10。 如果部件1和部件2的可改进比例均为30%，那么当部件3的可改进比例为多少时，系统总加速比才可以达到10？ 如果三个部件的可改进比例分别为30%、30%和20%，三个部件同时改进，那么系统中不可加速部分的执行时间在总执行时间中的比例是多少？

答案/提示： 把原总时间归一化。多部件 Amdahl 写成：S = 1 / ((1 - f1 - f2 - f3) + f1/s1 + f2/s2 + f3/s3)。 第1问列式：0.1 = (1 - 0.3 - 0.3 - f3) + 0.3/30 + 0.3/20 + f3/10，解得 f3 = 65/180 ≈ 0.36。 第2问改进后总时间为 0.3T/30 + 0.3T/20 + 0.2T/10 + 0.2T = 0.245T，不可加速部分占比 0.2T/0.245T ≈ 0.82。

【原题】CPU 时间与平均 CPI：比较两种设计

原题 1.4.2；原题截图：p.6；答案截图：p.7

FP 操作比例为25%，FP 操作平均 CPI = 4.0，其他指令平均 CPI = 1.33；FPSQR 操作比例为2%，FPSQR 的 CPI = 20。 有两种设计方案：方案一把 FPSQR 操作的 CPI 减为2；方案二把所有 FP 操作的 CPI 减为2。试利用 CPU 性能公式比较两种设计。

答案/提示： 先用改进前平均 CPI 反推出非 FPSQR 的 FP 指令 CPI：4 x 0.25 + 1.33 x 0.75 = 20 x 0.02 + CPI非FPSQR x 0.98，得 CPI非FPSQR = 80/49。 方案一：CPI = 2 x 0.02 + (80/49) x 0.98 = 1.64。 方案二：CPI = 2 x 0.25 + 1.33 x 0.75 ≈ 1.50。CPI 更小，所以方案二更好。

自测问题

合上正文后先试着口答，再展开看答案。能把这些问题讲清楚，本章主线基本就稳了。

响应时间适合评价单个任务完成快慢；吞吐率适合评价服务器或批处理系统单位时间完成多少任务。

IC 受算法/编译器/ISA 影响，CPI 受微结构和存储停顿影响，时钟周期受实现工艺和关键路径影响。

不同 ISA 的一条指令工作量不同，MIPS 只看每秒执行指令数，可能掩盖真实执行时间。

把总时间归一为 1，不可改进部分保留，每个可改进部分写成 fi/si，改进后总时间取倒数。

多数现代多处理器属于 MIMD，即多指令流、多数据流。

3. 流水线技术

把流水线当成“时空图 + 三类冲突 + 性能公式 + 非线性调度”来复习。考试重点通常不是背概念，而是会画、会算、会判断停顿。

1. 流水线分类

分类角度	类型	含义
功能连接是否固定	静态流水线	同一时刻各段只能按一种功能连接方式工作。
功能连接是否固定	动态流水线	同一时刻不同段可以按不同功能连接方式工作，控制更复杂。
是否有反馈回路	线性流水线	任务依次经过各段，每段最多经过一次。
是否有反馈回路	非线性流水线	存在反馈回路，一个任务可能多次使用某些功能段。

2. 流水线性能三件套

流水线性能题先判断是否是简单线性流水线。如果各段时间相等，用等时公式；如果各段时间不等，用最长段作为节拍。

等时线性流水线完成 n 个任务的时间：$Tk = (k + n - 1) \times \Delta t$

不等时线性流水线完成 n 个任务的时间：$Tk = \Sigma ti + (n - 1) \times \max(t1, t2, ..., tk)$

吞吐率 $TP = n / Tk$；加速比 S = 顺序时间 / 流水时间；效率 E = n 个任务实际占用的时空面积 / 流水线总时空面积

3. MIPS 五段流水线

阶段	名称	主要工作
IF	取指	用 PC 访问指令存储器，取出指令，通常 PC + 4。
ID	译码/读寄存器	译码、读寄存器、生成控制信号，部分实现会在此处理分支。
EX	执行	ALU 运算、地址计算、条件判断。
MEM	访存	Load/Store 访问数据存储器。
WB	写回	把结果写回寄存器堆。

画时空图时要把每条指令在每个周期处于哪个阶段标清楚，再根据相关关系判断是否插入气泡、重定向或清空。

4. 相关与冲突

相关 dependency

指令之间存在语义关系，包括数据相关、名相关、控制相关。相关是“原因”。

冲突 hazard

由于相关或资源不足，下一条指令无法在预定周期执行。冲突是“流水线中的后果”。

冲突类型	来源	解决思路
结构冲突	同一周期多条指令争用同一硬件资源。	资源重复、分离指令/数据存储、插入停顿。
数据冲突	指令之间读写寄存器或内存的顺序受约束。	数据重定向、插入气泡、编译器调度、寄存器重命名。
控制冲突	分支、跳转改变 PC，后续取指方向不确定。	分支预测、分支延迟槽、提前判断、错误路径清空。

5. 数据冲突：RAW、WAR、WAW

名称	必须保持的顺序	直觉记忆	MIPS 五段流水线中
RAW 写后读	前一条先写，后一条再读。	真正的数据依赖，最常见。	会发生，常用重定向解决。
WAR 读后写	前一条先读，后一条再写。	名相关，不是真数据流。	读在 ID、写在 WB，通常不会发生。
WAW 写后写	前一条先写，后一条再写。	输出相关。	五段顺序写回时通常不发生，乱序/多功能部件中要注意。

重定向一般从 EX/MEM、MEM/WB 流水寄存器把结果送回 ALU 输入端。但 Load-Use 情况下，数据到 MEM 后才可用，紧接着的下一条指令 EX 阶段太早，通常仍需插入 1 个气泡。

6. 控制冲突与分支处理

- 分支越早确定，错误取指造成的损失越小。
- 静态预测在编译或固定规则下完成，如总预测不跳转。
- 动态预测根据运行历史调整，和后面的 BHT、BTB 内容衔接。
- 分支延迟槽是把分支后必定会执行或可安全执行的指令放进延迟周期。

7. 非线性流水线调度

非线性流水线因为存在反馈，同一任务可能重复使用某段，所以不能随便每周期输入新任务。调度目标是在不发生功能段冲突的前提下，让平均启动间隔尽量小。

资料例题中的禁止表为 $F = \{1, 5, 6, 8\}$，对应初始冲突向量 $C0 = 10110001$。考试遇到类似题，按这四步机械推进即可。

8. 考前抓手

9. 公式与习题精讲

线性流水线公式总表

场景	公式	说明
等时 k 段，n 个任务	$T_k = (k + n - 1)\Delta t$	$k\Delta t$ 是第一个任务完成时间，后续每 Δt 完成一个。
不等时 k 段	$T_k = \Sigma t_i + (n - 1)\max(t_i)$	节拍由最慢段决定。
吞吐率	$TP = n / T_k$	n 趋于无穷大时，最大 $TP = 1/\Delta t$ 或 $1/\max(t_i)$。
加速比	$S = T_{顺序} / T_{流水}$	等时顺序时间通常为 $nk\Delta t$。
效率	$E =$ 有效时空面积 / 总时空面积	等时理想线性流水线 $E = n / (k + n - 1)$。

例题模板：等时流水线

5 段流水线，每段 10ns，处理 20 个任务。

$T_k = (5 + 20 - 1) \times 10ns = 240ns$

$TP = 20 / 240ns = 83.33$ M任务/s

$T_{顺序} = 20 \times 5 \times 10ns = 1000ns$，$S = 1000/240 = 4.17$

$E = 20 / (5 + 20 - 1) = 0.833$

MIPS 五段流水线时空图题

非线性流水线调度例题

资料例题预约表得到禁止表 $F = \{1, 5, 6, 8\}$，初始冲突向量 $C0 = 10110001$。

10. 原题练习区

【原题】静态多功能流水线：乘加表达式

例 3.1；原题截图：p.5

要在图示静态流水线上计算 $\Pi(i=1..4)(A_i + B_i)$，流水线输出可以直接返回输入端或暂存于相应流水寄存器中。试计算吞吐率、加速比和效率。
练习重点：先安排 4 次加法，再安排乘法树，最后由时空图统计总时间和阴影面积。

答案/提示：原资料时空图给出 $18\Delta t$ 内输出 7 个结果，所以 $TP = 7/(18\Delta t)$。不用流水线时总时间为 $(4 \times 6 + 3 \times 4)\Delta t = 36\Delta t$，所以 $S = 36/18 = 2$。效率按阴影面积与总时空面积计算：$E = (4 \times 6 + 3 \times 4)/(8 \times 18) = 0.25$。

【原题】动态多功能流水线：求和乘积

例 3.2；原题截图：p.6

一条动态多功能流水线由 5 段组成：加法使用 1、3、4、5 段；乘法使用 1、2、5 段；第 4 段时间为 $2\Delta t$，其余各段时间均为 Δt，且输出可直接返回输入端或暂存于流水寄存器中。若计算 $\Sigma(i=1..4)(A_i \times B_i)$，试计算吞吐率、加速比和效率。

答案/提示：原资料时空图给出 $16\Delta t$ 内输出 7 个结果，所以 $TP = 7/(16\Delta t)$。不用流水线时，一次求积需 $3\Delta t$，一次求和需 $5\Delta t$，总时间为 $(4 \times 3 + 3 \times 5)\Delta t = 27\Delta t$。加速比 $S = 27/16 \approx 1.69$；效率 $E = (4 \times 3 + 3 \times 5)/(5 \times 16) \approx 0.338$。

【原题】MIPS 五段流水线：循环、定向与调度

例 3.11；原题截图：p.10；解答截图：p.11；调度截图：p.12

在 MIPS 五段流水线中运行如下循环，R3 初始值为 R2 + 396；假设所有存储器访问都命中 Cache。 LOOP: LW R1, 0(R2) ADDI R1, R1, #1 SW 0(R2), R1 ADDI R2, R2, #4 SUB R4, R3, R2 BNZ R4, LOOP
没有任何定向硬件支持时，画出循环流水线时空图，并求总周期数。
有正常定向路径且采用“ 预测分支失败 ”策略时，画出时空图，并求总周期数。
有正常定向路径时，对循环指令重新排序，不能增加指令条数，画出时空图并计算总周期数。

答案/提示： 循环次数为 396/4 = 99。无定向时，原资料给出每轮 15 个时钟周期，总周期数为 15 x 99 + 3 = 1488。有定向且预测分支失败时，每轮 9 个时钟周期，总周期数为 9 x 99 + 3 = 894。 调度思路：把独立指令提前，尽量让 SW 填入分支延迟位置；具体重排序列见原资料 p.12。

【原题】预约表、禁止表与最优调度

非线性流水线调度；预约表：p.14；解答：p.15

给定 5 个功能段的非线性流水线预约表：S1 在时间 1、9 使用；S2 在时间 2、3、8 使用；S3 在时间 4 使用；S4 在时间 5、6 使用；S5 在时间 7、8 使用。 求禁止表 F、初始冲突向量 C0，画状态转移图，并给出平均启动间隔最小的调度方案。

答案/提示： 同一功能段内使用时间差构成禁止表：F = {1, 5, 6, 8}。 初始冲突向量 C0 = 10110001。根据状态图，方案 (3,4) 的平均启动间隔为 3.5 个时钟周期，是较优方案。

自测问题

合上正文后先试着口答，再展开看答案。能把这些问题讲清楚，本章主线基本就稳了。

相关是指令之间的语义依赖或名字关系；冲突是由于相关或资源不足导致流水线不能按预定周期推进。

后一条指令需要读取前一条尚未写回的结果，是真正的数据流依赖，可用重定向和停顿处理。

Load 的数据通常到 MEM 末尾才可用，紧随其后的使用者在 EX 阶段需要该数据，时间上来不及。

根据预约表找禁止向量，构造冲突向量和状态图，再选平均启动间隔最小且合法的循环。

启动/排空、停顿、功能段不均衡和冲突都会让部分功能段处于空闲状态。

4. 指令级并行

这一章继续回答“流水线之后还能怎么降 CPI”：动态分支预测减少控制停顿，多指令流出让理想 CPI 小于 1。

1. 指令级并行的核心

指令级并行 ILP 指指令序列中潜在的并行性。当指令之间没有必须保持的相关关系时，可以让它们重叠执行或同时流出。

流水线实际 CPI = 理想 CPI + 结构相关停顿 + 控制相关停顿 + 数据相关停顿

本章的很多技术都可以放进这个公式中理解：分支预测减少控制停顿，多指令流出降低理想 CPI，动态调度减少数据相关导致的等待。

2. BHT：预测分支是否成功

BHT，即分支历史表，用来记录分支指令过去一次或多次的执行结果，从而预测下一次分支是否成功。

预测器	机制	特点
1 位预测	只记录最近一次结果，成功则预测成功，不成功则预测不成功。	简单，但循环边界容易连续错两次。
2 位预测	用 2 位饱和计数器表示强/弱成功、强/弱不成功。	需要连续两次预测错误才改变方向，更稳定。
n 位预测	计数器范围为 0 到 2^n - 1，成功加 1，不成功减 1。	计数值大于等于中间阈值则预测成功。

3. BTB：预测分支目标地址

BTB，即分支目标缓冲器，用分支指令地址作为标识，保存该分支成功时的目标地址。取指阶段一旦 PC 与 BTB 中的标识匹配，就可以直接给出目标地址，减少等待。

BHT 回答

这条分支大概率跳不跳？

BTB 回答

如果跳，目标地址在哪里？

原版 BTB 命中时往往默认分支成功；改进方案是在 BTB 中增设至少 2 位的 BHT 字段，命中后仍要判断是否真的预测成功。

4. 分支目标指令缓冲器

分支目标指令缓冲器不只保存目标地址，还保存分支目标处的若干条指令。这样在取指阶段就能直接提供目标指令，对多流出处理器尤其有帮助。

- 无条件分支可能达到零延迟。
- 部分条件分支在预测准确时也可以接近零延迟。

- BHT 改进和分支目标指令缓冲器是相互独立的，可以同时使用。

5. 多指令流出技术

技术	流出方式	调度责任	复习关键词
超标量 Superscalar	每周期可流出多条，条数不固定但有上限。	可静态调度，也可硬件动态调度。	硬件复杂，灵活。
超长指令字 VLIW	每周期流出固定数量，打包成长指令或指令包。	主要由编译器静态完成。	硬件相对简单，依赖编译器。
超流水线 Superpipeline	把流水段继续细分，一个时钟周期内分时流出多条。	仍沿流水线节拍推进。	不是同时流出，而是每隔 1/n 周期流出一条。

6. 动态调度补充理解

高性能处理器还会使用乱序执行、寄存器重命名、重排序缓冲 ROB 等机制。它们的目标是在保持程序最终语义正确的前提下，让已经准备好的指令先执行。

- 寄存器重命名可以消除 WAR、WAW 这类名相关。
- ROB 用于按程序顺序提交结果，保证异常精确性。
- 动态调度适合运行时才能确定延迟的情况，如 Cache miss。

7. 考前抓手

8. 公式与习题精讲

ILP 性能公式

CPI实际 = CPI理想 + 结构相关停顿 + 控制相关停顿 + 数据相关停顿

m 流出处理机的理想 CPI ≈ 1 / m

理想 CPI 小于 1 并不代表实际一定小于 1，因为分支错误、访存失效、相关等待都会把 CPI 加回来。

BHT 状态题

预测器	状态更新	题型做法
1 位 BHT	上次成功则预测成功，上次失败则预测失败。	按分支实际序列逐次改位。
2 位 BHT	饱和计数器：成功加 1，失败减 1。	计数器高半区预测成功，低半区预测失败。
n 位 BHT	范围 0 到 2^n - 1。	值 >= 2^(n-1) 预测成功，否则预测失败。

2 位预测器例题模板

若状态编码为 00 强不成功、01 弱不成功、10 弱成功、11 强成功，初始 10，实际序列为成功、成功、失败、成功：

10 -> 11 -> 11 -> 10 -> 11

只有当连续错误把状态推过中线，预测方向才会改变。

BTB 延迟题

- BTB 命中且预测成功：取指阶段即可得到目标地址，分支开销最小。
- BTB 命中但实际不成功：需要删除或修改项，并清空错误路径。
- BTB 不命中但实际成功：要等分支计算出目标地址，再补入 BTB。
- 增设 BHT 后，BTB 不再“命中就默认成功”，而是同时判断是否跳转。

多指令流出比较题

题目问法	回答抓手
为什么超标量复杂？	硬件要动态判断相关、选择可流出指令、分配执行部件。
为什么 VLIW 依赖编译器？	并行性在指令包中显式表示，编译器静态调度。
超流水线和超标量区别？	超流水线是分时流出，超标量是同一周期多条同时流出。

9. 原题练习区

【原题】分支预测技术：目标缓冲与固定延迟比较

习题 5.8；原题截图：p.5

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。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 90%，预测精度为 90%，分支频率为 15%，没有分支的基本 CPI 为 1。求程序执行的 CPI。与固定 2 个时钟周期延迟的分支处理相比，哪种更快？

答案/提示：额外开销 = 15% x (90% x 10% x 4 + 10% x 3) = 0.099，所以 CPI = 1.099。固定 2 周期延迟的 CPI = 1 + 15% x 2 = 1.3，因此分支目标缓冲方案更快。

【原题】BTB：无条件转移进入目标缓冲

BTB 例题；原题截图：p.6

假定分支目标缓冲的命中率为 90%，程序中无条件转移指令为 5%，其它指令 CPI 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，问 CPI 是多少？假定原来的 CPI 为 1.1。

答案/提示：不采用 BTB 时：1 + 5% x L = 1.1，得 L = 2。采用 BTB 后，只有 10% 未命中仍付出 2 周期开销：CPI = 1 + 5% x 10% x 2 = 1.01。

【原题】超标量、VLIW、超流水线：时空图与加速比

习题 5.11；原题截图：p.6；续页：p.7

设指令流水线由取指令、分析指令和执行指令 3 个部件构成，每个部件需 Δt。连续执行 12 条指令，分别画出 ILP 为 4 的超标量处理机、超长指令字处理机和超流水线处理机的时空图，并分别计算相对于标量流水处理机的加速比。

答案/提示： 标量流水处理机：Tk = (3 + 12 - 1) Δt = 14 Δt。 超标量处理机：ILP = 4，12 条指令组成 3 组，Tk = (3 + 3 - 1) Δt = 5 Δt，S = 14/5 = 2.8。 超长指令字处理机：同样打包成 3 条长指令，Tk = 5 Δt，S = 2.8。 超流水线处理机：每 1/4 个时钟周期启动一条指令，执行 12 条指令需 5.75 Δt，S = 14/5.75 ≈ 2.43。

自测问题

合上正文后先试着口答，再展开看答案。能把这些问题讲清楚，本章主线基本就稳了。

它利用运行时历史信息更新预测状态，可以适应循环和程序实际行为。

BHT 预测分支是否成功；BTB 缓存分支目标地址，让预测成功时更快取到目标指令。

它需要连续两次相反结果才改变强预测方向，能减少循环末尾一次失误造成的来回翻转。

超标量主要靠硬件动态发现并发射多条指令；VLIW 主要靠编译器静态打包固定数量指令。

因为一个周期可以流出多条指令，若每周期平均完成 m 条，理想 CPI 约为 1/m。

5. 存储系统

本章内容最多，建议用一条线串起来：为什么要层次化存储，Cache 四问怎么回答，性能公式怎么套，最后主存带宽怎么优化。

1. 为什么需要存储层次

理想存储器希望容量大、速度快、价格低，但现实中三者互相制约。层次化存储用小而快的上层存储保存热点数据，用大而慢的下层存储提供容量。

时间局部性

刚访问过的数据或指令，不久后很可能再次访问。

空间局部性

访问某地址后，很可能继续访问附近地址。

块传送

Cache 和主存通常按块交换数据，正是为了利用空间局部性。

2. Cache 四个设计问题

问题	名称	常见答案
主存块调入 Cache 时放哪里？	映像规则	直接映像、全相联映像、组相联映像。
访问时怎么找到目标块？	查找算法	在候选位置比较标记，可顺序或并行查找。
Cache 满了替换哪块？	替换算法	随机、FIFO、LRU。
写访问怎么处理？	写策略	写直达、写回；按写分配、不按写分配。

3. 映像规则与地址字段

映像方式	优点	缺点	地址直觉
直接映像	硬件简单、命中时间短。	冲突失效率高。	块号对 Cache 行数取模得到行号。
全相联映像	冲突少，替换灵活。	需要比较所有标记，硬件复杂。	没有固定索引，主要靠标记比较。
组相联映像	折中方案。	相联度提高会增加比较器和命中时间。	先定位组，再在组内比较路。

块内偏移位数 = log2(块大小 Byte 数)

组索引位数 = log2(Cache 组数)

标记位数 = 地址总位数 - 组索引位数 - 块内偏移位数

4. 替换算法与写策略

替换算法

随机法：实现简单。

FIFO：替换最早调入的块。

LRU：替换最久未被访问的块，通常失效率较低但实现复杂。

写策略组合

写命中：写直达或写回。

写不命中：按写分配或不按写分配。

常见搭配：写回 + 按写分配；写直达 + 不按写分配。

写直达一致性好但写流量大，可用写缓冲器减少 CPU
等待；写回速度快但需要污染位，替换脏块时才写回下一级。

5. Cache 性能公式

AMAT = 命中时间 + 失效率 × 失效开销

CPU 时间 = IC × (基础 CPI + 存储器停顿周期数) × 时钟周期时间

平均访问时间衡量存储访问本身，但最终比较系统性能时常常要回到 CPU 时间。

两级 Cache

L1 失效开销 = L2 命中时间 + L2 局部失效率 × L2 失效开销

全局失效率 = 该级失效次数 / 所有访存次数；局部失效率 = 该级失效次数 / 到达该级的访存次数

6. 混合 Cache 与分离 Cache

结构	含义	优点	缺点
混合 Cache	指令和数据共用同一个 Cache。	空间利用更灵活，硬件相对简单。	取指和访数据可能争用同一 Cache 端口。
分离 Cache	指令 Cache 和数据 Cache 分开。	取指和访数据可并行，减少结构冲突。	可能出现指令 Cache 空闲而数据 Cache 紧张的空间不均衡。

分离 Cache 平均失效率 = 指令访问比例 × 指令 Cache 失效率 + 数据访问比例 × 数据 Cache 失效率

题目若给“ 75% 访存为取指，25% 为数据访问 ”，就按比例加权计算分离 Cache
的平均失效率或平均访问时间。

7. 三类失效与优化

失效类型	原因	优化思路
强制性失效	第一次访问某块，Cache 中一定没有。	增大块大小、预取。

失效类型	原因	优化思路
容量失效	工作集大于 Cache 容量，块被换出后又访问。	增加 Cache 容量，改善程序局部性。
冲突失效	多个常用块映射到同一组或同一行。	提高相联度、Victim Cache、伪相联 Cache。

- 块大小增大最初会降低失效率，但过大时块数减少，冲突和失效开销可能上升。
- 2:1 经验规则：容量为 N 的直接映像 Cache 失效率，常接近容量为 N/2 的两路组相联 Cache。
- Victim Cache 对小容量直接映像数据 Cache 的冲突失效尤其有效。

8. 减少失效开销与命中时间

读失效优先于写

检查写缓冲器，避免读请求被等待写回拖慢。

子块放置

标记仍按大块，传送按子块，减少失效开销。

请求字优先

先把 CPU 立即需要的字送回来，让 CPU 尽快继续执行。

非阻塞 Cache

Cache miss 时允许后续命中继续处理，隐藏部分等待。

减少命中时间的核心是小而简单：L1 Cache
容量不宜过大，结构不宜过复杂，否则可能限制处理器时钟频率。

9. 主存：延迟与带宽

主存位于 Cache 和辅存之间，性能主要看延迟和带宽。Cache
不命中开销常常由送地址、访问时间、数据传输时间构成。

技术	作用	题目里的体现
增加存储器宽度	一次传更多数据，提高带宽。	同一 Cache 块需要的传输次数减少。
多体交叉存储器	连续地址分布在不同存储体，可并行访问。	失效开销可写成送地址 + 访问时间 + 块大小 × 传送时间。
独立存储体	多个体各自有控制线路，支持多个独立访存。	要注意体冲突和地址映射。

10. 考前抓手

11. 公式总表与变量含义

题型	公式	变量提醒
地址拆分	块内偏移 = log2(块大小)；组索引 = log2(组数)；标记 = 地址位数 - 组索引 - 块内偏移	块大小通常按 Byte 算，注意字地址/字节地址。
平均访存时间	AMAT = 命中时间 + 失效率 x 失效开销	失效率用小数，不要把百分数直接代入。
分离 Cache	AMAT = Pi(Hi + MRiPi) + Pd(Hd + MRdPd)	Pi/Pd 是指令/数据访问比例。
两级 Cache	AMAT = H1 + MR1(H2 + MR2P2)	MR2 是 L2 局部失效率，若题目给全局失效率要先换算。
存储器停顿	停顿周期/指令 = 访存次数/指令 x 失效率 x 失效开销	取指本身也算访存，Load/Store 还会增加数据访存。
CPU 时间	CPU 时间 = IC x (基础 CPI + 存储停顿周期/指令) x 时钟周期	比较系统性能时最终回到 CPU 时间。

局部失效率与全局失效率

局部失效率 = 该级 Cache 失效次数 / 到达该级 Cache 的访问次数

全局失效率 = 该级 Cache 失效次数 / 所有访存次数

L2 全局失效率 = L1 失效率 x L2 局部失效率

评价第二级 Cache 时，更应该看全局失效率；计算 L1 失效开销时，用 L2 局部失效率。

12. 习题套路精讲

套路一：平均访存时间再转 CPU 时间

CPU 时间 = IC x (CPI命中 + 每条指令平均访存次数 x MR x MP) x ClockCycle

套路二：两级 Cache

L1 失效开销 = H2 + MR2(local) x P2

AMAT = H1 + MR1 x (H2 + MR2(local) x P2)

例：L1 访问 1000 次失效 40 次，L2 失效 20 次，则 L1 失效率 = 40/1000 = 4%，L2 局部失效率 = 20/40 = 50%，L2 全局失效率 = 20/1000 = 2%。

套路三：比较 L2 相联度

资料习题中直接映像 L2：H2 = 10 周期，MR2 = 25%，P2 = 50 周期。

L1 失效开销(直接映像 L2) = 10 + 25% x 50 = 22.5 周期

两路组相联 L2：局部失效率降到 20%，但命中时间增加 0.1 个 CPU 周期。

L1 失效开销(两路 L2) = 10.1 + 20% x 50 = 20.1 周期

结论：要同时看命中时间增加和失效率下降，不能只看其中一个。

套路四：伪相联 Cache

伪命中率 = 直接映像失效率 - 两路组相联失效率

AMAT伪相联 = H1 + (MR1 - MR2) x 伪命中开销 + MR2 x 失效开销

直接映像第一次没命中、另一区命中叫伪命中；两路也找不到才是真失效。

套路五：预取

AMAT预取 = 命中时间 + MR x 预取命中率 x 预取开销 + MR x (1 - 预取命中率) x 失效开销

预取命中率表示“ 本来会失效的访问中，有多少被预取提前解决 ”。

套路六：主存带宽和失效开销

结构	失效开销模型
普通窄主存	块大小 x (送地址 + 访问时间 + 传送时间)
多体交叉	送地址 + 访问时间 + 块大小 x 传送时间
总线/存储器加宽 r 倍	(块大小/r) x (送地址 + 访问时间 + 传送时间)

套路七：脏块写回 + TLB

如果每次 Cache 失效一定读入一个块，且有 50%
概率需要脏块写回，则平均每次失效的数据交换次数为：

0.5 x 1 + 0.5 x 2 = 1.5

若主存延迟 40 周期，块大小 32B，每次传 4B，TLB 不命中率 0.2%，TLB 不命中开销 20 周期：

失效开销 = 1.5 x (40 + 32/4 + 0.2% x 20) = 72.06 周期

CPI = 基础 CPI + 平均每条指令访存次数 x MR x 失效开销

13. 原题练习区

【原题】Cache 对 CPU 时间的影响

例 7.1；原题截图：p.19

用一个和 Alpha AXP 类似的机器作为例子：Cache 不命中开销为 50
个时钟周期；不考虑存储器停顿时，所有指令执行时间都是 2.0 个时钟周期；访问 Cache 的不命中率为 2%；平均每条指令访存 1.33 次。试分析 Cache 对性能的影响。

答案/提示： CPU 时间 = IC x (CPIexecution + 每条指令平均访存次数 x 不命中率 x 不命中开销) x 时钟周期。 代入得 CPU 时间 = IC x (2.0 + 1.33 x 2% x 50) x 时钟周期 = IC x 3.33 x 时钟周期。

【原题】直接映像 Cache 与两路组相联 Cache

例 7.2；原题截图：p.20；解答：p.21

比较 64KB、32B 块大小的直接映像 Cache 和两路组相联 Cache。理想 Cache CPI 为 2.0，时钟周期 2ns，平均每条指令访存 1.3 次。组相联使 CPU 时钟周期增至原来的 1.10 倍；两种结构不命中开销均为 70ns；命中时间为 1 个时钟周期；64KB 直接映像不命中率 1.4%，两路组相联不命中率 1.0%。问平均访存时间和 CPU 性能如何？

答案/提示： 平均访存时间：直接映像 = 2.0 + 0.014 x 70 = 2.98ns；两路组相联 = 2.0 x 1.10 + 0.010 x 70 = 2.90ns。 CPU 时间：直接映像约为 5.27 x IC；两路组相联约为 5.31 x IC。虽然两路组相联 AMAT 更低，但 CPU 时钟周期变长，所以按 CPU 时间直接映像略好。

【原题】两级 Cache：局部失效率、全局失效率与停顿

例 7.3；原题截图：p.21；解答：p.22

在 1000 次访存中，L1 不命中 40 次，L2 不命中 20 次。求各种局部不命中率和全局不命中率。 L2 命中时间为 10 个时钟周期，L2 不命中开销为 100 个时钟周期，L1 命中时间为 1 个时钟周期，平均每条指令访存 1.5 次，不考虑写操作影响。求平均访存时间和每条指令平均停顿周期。

答案/提示： L1 不命中率 = 40/1000 = 4%；L2 局部不命中率 = 20/40 = 50%；L2 全局不命中率 = 20/1000 = 2%。 AMAT = 1 + 4% x (10 + 50% x 100) = 3.4 个时钟周期。 每次访存停顿 = 3.4 - 1.0 = 2.4 周期；每条指令平均停顿 = 2.4 x 1.5 = 3.6 周期。

【原题】第二级 Cache 相联度对 L1 不命中开销的影响

例 7.4；原题截图：p.22；解答：p.23

第二级 Cache 数据：直接映像命中时间 10 周期；两路组相联命中时间增加 0.1 周期，即 10.1 周期；直接映像局部不命中率 25%；两路组相联局部不命中率 20%；不命中开销 50 周期。试问第二级 Cache 相联度对第一级 Cache 不命中开销有什么影响？

答案/提示： 直接映像 L2：L1 不命中开销 = 10 + 25% x 50 = 22.5 周期。 两路组相联 L2：L1 不命中开销 = 10.1 + 20% x 50 = 20.1 周期。若机器要求命中时间取整，可讨论取 10 或 11 周期两种情况。

【原题】分离 Cache 与混合 Cache

例 5.1；原题截图：p.23；解答：p.24

假设 Cache 命中时间为 1 个时钟周期，失效开销为 50 个时钟周期。在混合 Cache 中一次 load/store 访问 Cache 的命中时间都要增加 1 个时钟周期。根据表中失效率，比较指令 Cache 和数据 Cache 容量均为 16KB 的分离 Cache 与容量为 32KB 的混合 Cache：哪种失效率更低？两种情况下平均访存时间分别是多少？

答案/提示： 约 75% 访存为取指令，25% 为数据访问。16KB 分离 Cache 总失效率 = 75% x 0.64% + 25% x 6.47% = 2.10%。32KB 混合 Cache 失效率为 1.99%，失效率略低。 平均访存时间：分离 Cache = 75% x (1 + 0.64% x 50) + 25% x (1 + 6.47% x 50) = 2.05；混合 Cache = 75% x (1 + 1.99% x 50) + 25% x (1 + 1 + 1.99% x 50) = 2.24。

【原题】块大小、失效率和失效开销

例 5.4；原题截图：p.25

假定存储系统在延迟 40 个时钟周期后，每 2 个时钟周期能送出 16 个字节。即经过 42 个时钟周期可提供 16B，44 个时钟周期可提供 32B，依此类推。试问对表 5.6 中各种容量的 Cache，块大小分别为多少时平均访存时间最小？

答案/提示： 对每个容量和块大小计算：AMAT = 命中时间 + 失效率 x 失效开销。命中时间假设与块大小无关，为 1 个周期。 失效开销 = 40 + 2 x 块大小/16。然后逐格比较表中失效率对应的 AMAT，原资料中红色数值就是每列较优选择。

【原题】提高相联度是否一定更快？

例 5.5；原题截图：p.26

假定提高相联度会按下列比例增大处理器时钟周期：2 路 = 1.10 x 直接映像时钟周期，4 路 = 1.12 x，8 路 = 1.14 x。命中时间为 1 个时钟周期，直接映像下失效开销为 50 个时钟周期，且不必将失效开销取整。使用表 5.5 中的失效率，问当 Cache 多大时满足：8路平均访存时间 < 4路 < 2路 < 1路？

答案/提示： 分别计算：8路 AMAT = 1.14 + MR8 x 50；4路 AMAT = 1.12 + MR4 x 50；2路 AMAT = 1.10 + MR2 x 50；1路 AMAT = 1.00 + MR1 x 50。 逐个 Cache 容量代入表 5.5 失效率，找满足不等式的容量。

【原题】伪相联 Cache：直接映像、两路组相联和伪相联比较

例 5.6；原题截图：p.26

假设当在直接映像找到的位置没有发现匹配，而在另一个位置才找到数据，即伪命中，需要 2 个额外周期。仍用前一例数据，问当 Cache 容量分别为 2KB 和 128KB 时，直接映像、两路组相联和伪相联三种组织结构中哪一种速度最快？

答案/提示： 伪相联平均访存时间 = 直接映像命中时间 + (直接映像失效率 - 两路组相联失效率) x 伪命中开销 + 两路组相联失效率 x 真失效开销。 分别代入 2KB 和 128KB 的直接映像/两路组相联失效率比较。

自测问题

合上正文后先试着口答，再展开看答案。能把这些问题讲清楚，本章主线基本就稳了。

先算块内偏移位数 log2(块大小)，再算组索引位数 log2(组数)，剩余为标记位。

全相联冲突少但查找复杂；直接映像简单快但冲突多；组相联在二者之间折中。

写回命中时只改 Cache、替换脏块时再写主存；写直达每次写命中都同步写下一级。

局部失效率以到达 L2 的访问为分母；全局失效率以所有访存为分母，等于 L1 失效率乘 L2 局部失效率。

连续地址分布到不同存储体，各体可并行工作，从而在访存周期内传回更多数据。

6. I/O 外存系统

本章重点是可靠性与 RAID。复习时把 I/O 性能指标、可靠度公式、RAID 各级特点放在一起记，计算题就会清楚很多。

1. I/O 系统做什么

I/O 系统完成计算机和外界的信息交换，也为计算机提供大容量外部存储器。课程重点偏向存储 I/O 系统。

- I/O 系统包括 I/O 设备，以及 I/O 设备与处理机之间的连接。
- I/O 系统性能会影响 CPU 性能，性能不匹配时可能成为整个系统瓶颈。
- 评价参数包括连接特性、容量、响应时间、吞吐率、I/O 操作对 CPU 的打扰情况。
- 用户感受到的响应时间通常包含 I/O 响应时间和 CPU 处理时间。

2. 可靠性、可用性、可信性

概念	含义	常用指标
可靠性 Reliability	系统从某个初始参考点开始连续提供服务的能力。	MTTF；失效率 = 1 / MTTF。
可用性 Availability	系统正常工作时间在正常服务间隔中的比例。	MTTF / (MTTF + MTTR)。
可信性 Dependability	服务质量，即多大程度上可合理认为服务可靠。	通常不可直接度量。

$$\text{Availability} = \text{MTTF} / (\text{MTTF} + \text{MTTR})$$

3. 串联、并联与混联系统

串联系统

所有组件都正常，系统才正常。任意一个组件失效，系统失效。

$$Rs = R1 \times R2 \times \ldots \times Rn$$

并联系统

只要至少一个组件正常，系统就正常。全部组件失效，系统才失效。

$$Rs = 1 - \Pi(1 - Ri)$$

混联系统先判断结构：RAID 0+1 可理解为先条带再镜像，RAID 1+0 可理解为先镜像再条带。资料里用串并联系统、并串联系统公式来解释两者可靠性差异。

4. RAID 分级速查

级别	核心思想	性能/可靠性印象
RAID 0	条带化，无冗余。	性能高，可靠性差，任一盘坏可能丢数据。

级别	核心思想	性能/可靠性印象
RAID 1	镜像，数据有一份备份。	读快，写需写两份，可靠性高，成本高。
RAID 2	位交叉，汉明码校验。	概念性强，商业应用少。
RAID 3	位交叉，单独奇偶校验盘。	细粒度，适合大块连续传输。
RAID 4	块交叉，单独奇偶校验盘。	小写会集中访问校验盘，可能成为瓶颈。
RAID 5	块交叉，奇偶校验分布在所有盘。	消除 RAID4 单校验盘热点，常用。
RAID 6	P + Q 双校验。	可容忍两个磁盘故障，校验开销更大。
RAID 10	先镜像 RAID1，再条带 RAID0。	可靠性通常优于 RAID01，性能也较好。
RAID 01	先条带 RAID0，再镜像 RAID1。	条带组中一盘故障会影响整个组，容错较弱。

5. RAID 写操作直觉

RAID4/5 的小写操作常见计算是“ 读旧数据、读旧校验、写新数据、写新校验 ”，也就是 2 次读和 2 次写。原因是奇偶校验需要根据旧值和新值更新。

新校验 = 旧校验 xor 旧数据 xor 新数据

如果是整条带写，可以重新计算整条带校验，开销模型会不同。做题时先看题目说的是小写、整条带写还是磁盘故障恢复。

6. RAID 实现方式

方式	特点
软件方式	阵列管理由主机软件完成，成本低但占用主机资源。
阵列卡方式	RAID 管理固化在 I/O 控制卡上，减少主机负担。
子系统方式	独立外部存储子系统，可服务多种主机平台。

7. 考前抓手

8. 公式与习题精讲

可靠度公式

结构	公式	做题判断
串联系统	R = R1 x R2 x ... x Rn	任一部件坏，系统坏。
并联系统	R = 1 - (1 - R1)(1 - R2)...(1 - Rn)	全部部件坏，系统才坏。
同失效率串联	$\lambda s = \Sigma \lambda i$，MTTF = $1/\lambda s$	指数分布下可用。

结构	公式	做题判断
n 个相同部件并联	MTTF = (1/λ)(1 + 1/2 + ... + 1/n)	资料并联系统推导结果。
可用性	Availability = MTTF/(MTTF + MTTR)	MTTR 越小，可用性越高。

混联系统

设单元可靠度为 Ri(t)，每串 n 个单元，再并联 m 组，是串并联系统：

R(t) = 1 - {1 - [Ri(t)]^n}^m

每组 m 个并联单元，再串联 n 组，是并串联系统：

R(t) = {1 - [1 - Ri(t)]^m}^n

资料对应：RAID 0+1 是串并联系统，RAID 1+0 是并串联系统。

RAID 容量与小写

RAID	可用容量粗略公式	可靠性/写开销
RAID0	nD	无冗余，任一盘坏会影响数据。
RAID1	nD/2	镜像，读可并行，写两份。
RAID5	(n - 1)D	分布式奇偶校验，可容忍 1 盘故障。
RAID6	(n - 2)D	双校验，可容忍 2 盘故障。
RAID10	nD/2	先镜像再条带，通常比 RAID01 更可靠。

RAID4/5 小写：读旧数据 + 读旧校验 + 写新数据 + 写新校验 = 2 次读 + 2 次写

新校验 = 旧校验 xor 旧数据 xor 新数据

故障恢复题

单盘故障时，缺失数据可由同一条带的其他数据和校验异或恢复：

丢失盘数据 = 校验 xor 其他所有数据盘数据

9. 原题练习区

【原题】RAID10 镜像磁盘阵列的系统可靠度

RAID 可靠性习题；原题截图：p.11；解答：p.12

一个镜像磁盘冗余阵列由 4 个磁盘配置为 RAID10 级，其结构如图，采用双控制器 RC 结构，任何一个阵列控制器失效不影响系统工作。已知各部分可靠度为：阵列控制器 R1 = 0.9，通道适配器 R2 = 0.95，磁盘 R3 = 0.95。 写出系统可靠性的表达式。 画出系统可靠性框图。 计算 R 的数值，保留小数点后两位。

答案/提示： 可靠性模型：两个控制器并联、通道适配器串联、三组镜像磁盘串联。 表达式：R = [1 - (1 - R1)^2] x R2 x [1 - (1 - R3)^2]^3。 代入：R = [1 - (1 - 0.9)^2] x 0.95 x [1 - (1 - 0.95)^2]^3 ≈ 0.93。

自测问题

合上正文后先试着口答，再展开看答案。能把这些问题讲清楚，本章主线基本就稳了。

可靠性强调连续无故障服务能力，可用性强调正常服务时间比例，可信性描述服务质量信任程度。

串联可靠度相乘；并联可靠度等于 1 减去所有部件都失效的概率。

通过数据条带化提高并行性能，通过镜像或校验冗余提高可靠性。

需要读旧数据、读旧校验、写新数据、写新校验，用旧校验 xor 旧数据 xor 新数据得到新校验。

RAID10 先镜像再条带，每组镜像中只要保留一个盘即可；RAID01
一侧条带失效后容错能力下降更明显。

7. 互联网络

互联网络要同时会背概念、会算图指标、会做互连函数变换、会按 Omega 网络寻径。复习时把“拓扑 + 函数 + 寻径”分开整理。

1. 基本概念与指标

互联网络是计算机部件、节点或系统之间的连接，目标是在延迟、成本、能耗等约束下传输尽可能多的数据，避免成为瓶颈。

指标	含义	复习用途
网络规模 N	网络中结点个数。	确定可连接部件数量。
结点度 d	与结点相连的边数，包括入度和出度。	反映单结点连接复杂度。
结点距离	两结点之间最短路径长度。	用于寻径和延迟分析。
网络直径 D	任意两结点距离的最大值。	越小，最坏通信路径越短。
等分宽度 b	把网络切成两半所需切断的最少边数。	反映最大流量潜力。
对称性	从任意结点看拓扑结构都相同。	对称网络更容易分析和扩展。

2. 互连函数

互连函数描述输入端编号 x 会连接到哪个输出端
f(x)。它可以用输入输出对、连线图、循环表示法、二进制位变换等方式表示。

函数	变换规则	记忆方式
恒等函数	f(x) = x。	同号直连。
交换函数	二进制地址第 k 位取反。	按某一位配对交换。
均匀洗牌函数	二进制编号循环左移一位。	像洗牌一样交叉。
逆均匀洗牌	二进制编号循环右移一位。	洗牌函数的逆。
蝶式函数	最高位与最低位互换。	多级立方体网络基础。
反位序函数	二进制位序颠倒。	如 001 变 100。
移数函数/PM2I	编号按模 N 加减某个偏移。	环上平移。

3. 静态互连网络

静态网络的结点连接在运行中不变，适合用图来分析度、直径、等分宽度和最短路径。

网格/环/线性阵列

结构直观，适合考察结点距离和直径。

超立方体

N = 2^n 个结点，每个结点用 n 位二进制编号，相邻结点通常只差 1 位。

超立方体的 E-cube
寻径可以按源地址和目的地址不同的位逐位翻转，路径长度等于两者二进制编号的汉明距离。

4. 寻径

类型	含义	例子
确定性寻径	路径完全由源结点和目的结点地址决定，不看当前拥塞。	二维网格 X-Y 寻径，超立方体 E-cube 寻径。
自适应寻径	根据资源、拥塞或故障状态动态选择路径。	可避开拥塞结点，提高网络利用率。

5. 动态互连网络

网络	特点	代价
总线	结构简单、成本低。	每次只能支持有限传送，带宽窄，争用明显。
交叉开关	可同时建立多个无冲突连接，带宽和互连能力强。	n × n 需要 n^2 个交叉点，规模大时代价高。
多级互连网络 MIN	用多级小开关实现较丰富的置换。	可能阻塞，寻径和控制更复杂。

多级网络的控制方式包括级控制、单元控制、部分级控制；级间互连模式包括均匀洗牌、蝶式、多路洗牌、立方体连接等。

6. Omega 网络

Omega 网络是多级混洗-交换网络。N 个输入时有 log2N 级，每级 N/2 个 2×2 开关，级间采用均匀洗牌连接，每个开关可直送或交换。

7. 考前抓手

8. 公式与习题精讲

互连函数公式

Cube_i(x) = x xor 2^i

均匀洗牌：二进制循环左移 1 位；逆均匀洗牌：二进制循环右移 1 位

n 维超立方体：N = 2^n，结点度 = n，直径 = n，两点距离 = 汉明距离

函数	公式/操作	例子
交换函数 Cube_i	f(x) = x xor 2^i	8 个端口中，x=3(011)，i=0，则 f=2(010)。
均匀洗牌	二进制循环左移 1 位	abc -> bca。
逆均匀洗牌	二进制循环右移 1 位	abc -> cab。

函数	公式/操作	例子
蝶式函数	最高位与最低位互换	abcde -> ebcda。
反位序	二进制位序完全反转	abcde -> edcba。
移数/PM2I	f(x) = x ± 2^i mod N	环形编号上前后移动。

静态网络常用结论

网络	结点度	直径/距离	做题抓手
n 维超立方体，N=2^n	n	直径 n，两点距离为汉明距离。	源和目的二进制有几位不同，最短路就几步。
混洗交换网络	按题图判断	常见直径 2n - 1	资料例题中 2^5 结点直径为 2n - 1 = 9。
网格	内部结点通常为 4	曼哈顿距离	X-Y 寻径先横后竖。

Omega 网络寻径题

N 输入 Omega 网络有 log2N 级，每级 N/2 个 2x2 开关，总开关数为：

开关数 = (N/2)log2N

例题模板

N=8 时有 3 级，每级 4 个 2x2 开关。若请求 P6(110) 到 P0(000)，先写出源/目的二进制，再按目的位 0、0、0 逐级选择输出方向，并检查与其他请求是否争用同一开关输出。

9. 原题练习区

【原题】互连函数、混洗交换网络与移数网络

习题 9.9；原题截图：p.20；解答：p.21

设函数的自变量是十进制数表示的处理机编号。现有 32 台处理机，其编号为 0,1,2,...,31。
分别计算：Cube2(12)、σ(8)、β(9)、PM2I+3(28)、Cube0(σ(4))、σ(Cube0(18))。用 Cube0 和 σ 构成混洗交换网，每步只能使用 Cube0 和 σ 一次。网络直径是多少？从 5 号处理机发送数据到 7 号处理机，最短路径要经过几步？请列出经过的处理机编号。
采用移数网络构成互连网，网络直径是多少？结点度是多少？与 2 号处理机距离最近的是哪几个处理机？

答案/提示：第1问：Cube2(12)=8，σ(8)=16，β(9)=24，PM2I+3(28)=4，Cube0(σ(4))=9，σ(Cube0(18))=7。
混洗交换网络直径为 2n - 1 = 9。5 到 7 的最短路径为 6 步：00101B -> 01010B -> 10100B -> 01001B -> 10010B -> 10011B -> 00111B。移数网络直径为 floor(5/2)=3，结点度为 2n - 1 = 9。与 2 号处理机距离最近的是 13、15、21、23 号处理机。

【原题】N=8 三级 Omega 网络的广播连接

习题 9.13；原题截图：p.21

> 用一个 N=8 的三级 Omega 网络连接 8 个处理机 P0-P7。8 个处理机的输出端分别依次连接 Omega 的 8 个输入端 0-7，8 个处理机的输入端分别依次连接 Omega 的 8 个输出端 0-7。 如果处理机 P6 要把数据播送到处理机 P0-P4，处理机 P3 要把数据播送到处理机 P5-P7，那么 Omega 网络能否同时为它们的播送要求实现连接？画出实现播送的 Omega 网络开关状态图。

答案/提示： 按 Omega 网络寻径方法处理：逐级根据目的地址位决定 2x2 开关直连或交换，并检查同一开关同一输出端是否冲突。 原资料给出了可实现的开关状态图；做题时建议先分别标出 P6 到 P0-P4、P3 到 P5-P7 的路径，再合并检查。

自测问题

合上正文后先试着口答，再展开看答案。能把这些问题讲清楚，本章主线基本就稳了。

互连结构、开关元件和控制方式。结构决定路怎么连，开关决定怎么转发，控制方式决定怎么调度。

把两个结点编号写成二进制，最短距离等于它们的汉明距离。

N 个输入的 Omega 网络通常有 log2N 级，每级 N/2 个 2x2 开关。

静态网络连接固定；动态网络由交换开关组成，可按程序请求动态改变连接状态。

先写二进制源/目的地址，再按网络规则逐级选路，最后检查是否发生开关输出冲突。

8. 多处理器

多处理器不是简单地“核越多越快”。并行性比例、通信延迟、Cache 一致性和同步开销共同决定最终速度。

1. 单处理器瓶颈与分类

单处理器瓶颈来自性能提升边际效应降低：时钟频率受功耗和散热限制，指令级并行也有边际收益。多处理器通过多个处理器并行工作来继续提升性能。

分类	特点	关键词
Flynn 分类	现代多处理器大多属于 MIMD。	多指令流、多数据流。
共享全局地址空间	逻辑地址空间统一，物理上可能分布。	NUMA，远程访问延迟不均。
分布式独立地址空间	各结点地址空间独立，远程数据需显式传递。	消息传递，集群。

2. 两个整体挑战

并行性有限

程序中总有一部分难以并行，Amdahl 定律给出整体加速上限。

```
Speedup = 1 / ((1 - f) + f / s)
```

通信开销较大

远程访问、同步等待、缓存一致性事务都会增加额外周期，抵消并行收益。

```
实际 CPI = 本地 CPI + 通信/同步停顿周期
```

3. 通信机制

机制	如何通信	优缺点
共享存储器	通过 Load/Store 访问共享地址空间。	编程直观，但依赖 Cache 一致性协议。
消息传递	显式 Send/Receive。	硬件较简单、适合分布式系统，但编程复杂度更高。

4. Cache 一致性

多个处理器各自有私有 Cache，同一主存块可能有多个副本。某处理器写了自己的副本后，其他副本可能变旧，这就是一致性问题。

协议	核心思想	适用印象
监听式协议 Snooping	所有 Cache 监听总线请求，发现相关块就改变本地状态。	广播式，适合较小规模共享总线。
写作废	写共享块时让其他副本无效。	MSI 属于典型写作废协议。
写更新	写共享块时广播新值给其他副本。	延迟低但带宽消耗大。

协议	核心思想	适用印象
目录式协议	目录记录每个块被哪些处理器缓存，按目录点对点协调。	减少广播，适合大规模系统。

5. MSI 协议

状态	含义	关键词
M Modified	唯一最新副本，尚未写回主存。	独占、脏、替换要写回。
S Shared	可有多个副本，与主存一致。	共享、干净。
I Invalid	该副本无效，不能使用。	失效、需重新取块。

总线事务重点记三个：读缺失 RdMiss、写缺失 WtMiss、作废 Invalidate。状态转换题要先判断本地读写还是总线事务，再改状态。

6. 同步机制

Cache 一致性保证副本值一致，同步机制保证共享资源访问顺序正确。同步一般用硬件原子操作构建。

原子操作	作用	记忆
Atomic Exchange	交换寄存器与内存值，可实现锁。	拿自己的值去换锁。
Test-and-Set	测试并设置，整个过程不可分割。	看一眼并立刻占住。
LL/SC	Load-Linked 后 Store-Conditional，中途无人修改才写成功。	先订阅，再提交。

旋转锁

不断检查锁变量直到可用。锁持有时间短时可用，竞争激烈会浪费周期和总线带宽。

栅栏同步

所有处理器到达同一点后才继续执行。可用 sense reversing 处理重复使用栅栏的问题。

7. 同步性能问题

- 总线争用：多个处理器争同一把锁，会产生大量一致性事务。
- 串行化开销：同步要求某些访问按顺序发生，限制并行。
- 局部副本优化：旋转锁可先在本地 Cache 副本上等待，减少反复访问总线。
- 资料例子提到 10 个处理器竞争锁可能产生 120 次事务和 12000 周期开销，说明同步不是免费的。

8. 考前抓手

9. 公式与习题精讲

并行加速比

```
Speedup = 1 / ((1 - f) + f / p)
```

f 是可并行部分比例，p 是处理器数量或并行部分理论加速比。即使 p 很大，Speedup 也不会超过 1/(1 - f)。

例题模板

若 90% 程序可并行，用 8 个处理器：

```
Speedup = 1 / (0.1 + 0.9/8) = 4.71
```

不是 8 倍，因为 10% 串行部分限制了上限。

远程访问与 CPI

```
CPI = 基础 CPI + 每条指令远程访问次数 x 远程访问额外延迟
```

```
CPI = 基础 CPI + 通信发生比例 x 通信开销周期
```

题目给“每 k 条指令发生一次远程访问”时，通信发生比例就是 1/k。若给本地和远程访问延迟差，要加的是额外延迟。

MSI 状态题

事件	常见状态变化	解释
本地读 I	I -> S，发 RdMiss	读缺失，取来共享副本。
本地写 I	I -> M，发 WtMiss	写缺失，需要独占并作废其他副本。
本地写 S	S -> M，发 Invalidate	已有共享副本，写前作废其他共享者。
监听到他人 RdMiss 且本地 M	M -> S，并提供/写回数据	他人要读，脏数据需要被看见。
监听到他人写缺失或作废且本地 S/M	S/M -> I	他人要写，本地副本失效。

同步开销题

- 旋转锁适合临界区很短的情况；等待时间长时，会浪费处理器周期。
- 多个处理器同时争锁会造成总线争用和大量一致性事务。
- 本地自旋优化：先在本地 Cache 副本上循环等待，锁释放时由一致性协议使副本失效，再重新竞争。
- 栅栏同步题重点看计数器是否能区分轮次；sense reversing 用每个处理器的私有 sense 解决复用问题。

```
同步后实际执行时间 = 理想并行执行时间 + 锁等待时间 + 一致性事务时间 + 栅栏等待时间
```

10. 原题练习区

【原题】Amdahl 定律：100 个处理器达到 80 倍加速

课程例 8.1；来源：第十章多处理器概念课件

```
假设想用 100 个处理器达到 80 的加速比，求原计算程序中串行部分最多可占多大的比例。
```

答案/提示： 令并行比例为 f：80 = 1 / ((1 - f) + f/100)。解得 f = 0.9975。 串行部分最多为 1 - f = 0.0025，即 0.25%。

【原题】远程访问延迟对 CPI 的影响

课程例 8.2；来源：第十章多处理器概念课件

一台 32 台处理器的多处理机，远程存储器访问时间为 200ns。除通信外，所有其它访问均命中局部存储器。发出远程请求时本处理器挂起。处理器时钟频率为 2GHz，基本 CPI 为 0.5。求没有远程访问与有 0.2% 指令需要远程访问时，前者比后者快多少。

答案/提示： 2GHz 的时钟周期为 0.5ns，远程访问开销为 200ns/0.5ns = 400 个时钟周期。 有远程访问时 CPI = 0.5 + 0.2% x 400 = 1.3。没有远程访问时 CPI = 0.5，所以前者速度是后者的 1.3/0.5 = 2.6 倍。

自测问题

合上正文后先试着口答，再展开看答案。能把这些问题讲清楚，本章主线基本就稳了。

前者多个处理器共享单一主存地址空间；后者存储器分布到各结点，访问本地和远程代价不同。

共享地址空间通过读写共享变量通信；消息传递通过显式发送/接收消息通信。

多个处理器私有 Cache 中同一内存块副本可能不一致，需要保证读写看到符合协议的最新值。

监听式靠总线广播和各 Cache 监听；目录式为每个内存块维护共享者/拥有者目录并点对点协调。

锁等待、总线争用、一致性事务和栅栏等待都会增加额外执行时间，抵消部分并行收益。