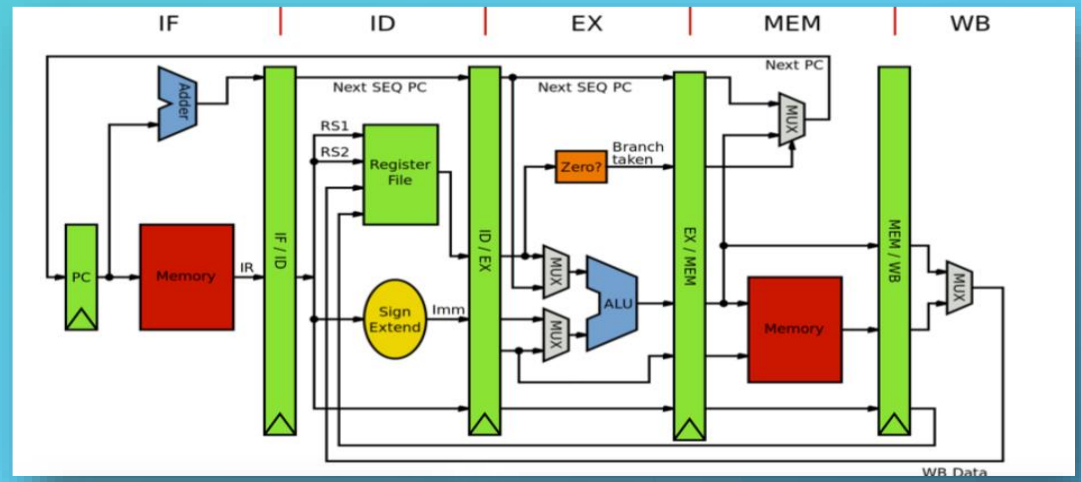
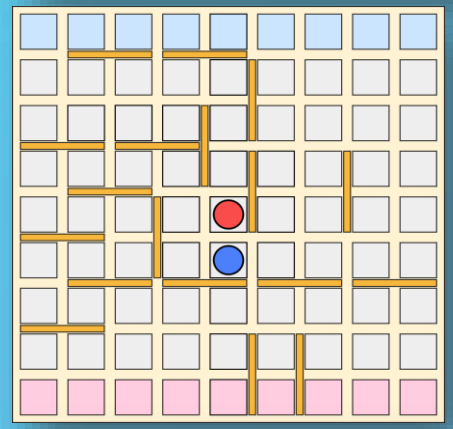
PPCA 2021

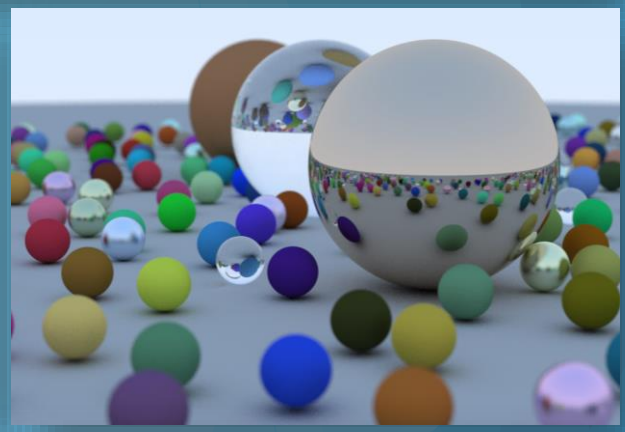
编程综合实践

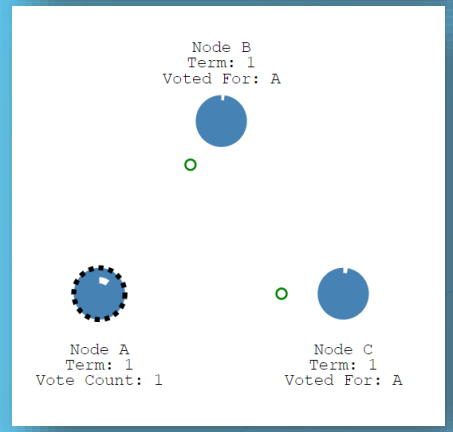
Principle and Practice of Computer Algorithms

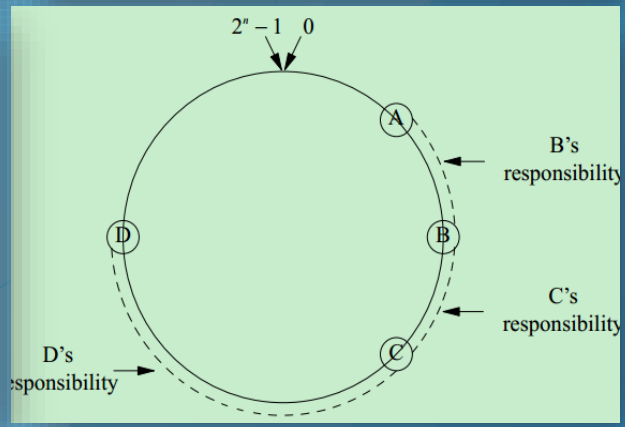
2020级ACM班 & 致远工科荣誉 2021.6.28











01 课程简介

课程简介

- 日期：
  - ACM：6月28日-8月7日， 周一到周五，共6周
  - 工科： 6月28日-7月24日，周一到周五，共4周
- 时间：
  - 除机考日外： 上午：9:00~12:00 下午：14:00~17:00
  - 机考日（第二周起每周四）：上午：9:00~11:00 下午：13:00~18:00
  - 机考可提前离场，机考后有讲题时间
- 地点：软件学院一楼5-104机房 （在一楼）
- 课程wiki： https://acm.sjtu.edu.cn/wiki/PPCA_2021

02 助教介绍

02 助教介绍

周聪
@cong258258

徐若凡
@Aqua

史涵雯
@Ivy

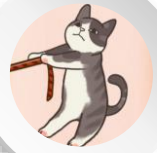
杨新宇
@malachite

张洪鑫
@Icefox

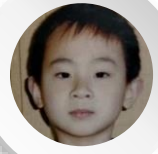
许振宇
@battlin

陈雪阳
@Anoxiacxy

黄臻
@xmhuangzhen



林春茹
@XHRlyb



闵乐钧
@aik2

02 助教介绍

周聪
@cong258258

徐若凡
@Aqua

RISC-V CPU模拟器：五级流水 或 托马斯洛Tomasulo乱序

史涵雯
@Ivy

Verilog 硬件描述语言作业

杨新宇
@malachite

Raft分布式一致性协议

张洪鑫
@Icefox

许振宇
@battlin

AI对战

陈雪阳
@Anoxiacxy

机考主管

黄臻
@xmhuangzhen

DHT 分布式哈希表

林春茹
@XHRlyb

闵乐钧
@aik2

Raytracer 光线追踪

03 项目安排与分数

03 项目安排与分数（暂定）

- 1. RISC-V CPU模拟器（第1周、第2周） 25%
- 2. Verilog硬件描述语言 小作业（第2周） 5%
- 3. 机考（第2周起，工科共3次，ACM班共5次） 30%
- 4. 分组项目四选一（四个项目均为工科第3周第4周，ACM班第3-6周） 40%
  - AI对战 （工科、ACM班可选）
  - Raytracer 光线追踪 （工科、ACM班可选）
  - DHT 分布式哈希表 （ACM班可选）
  - Raft 分布式一致性协议 （ACM班可选）
- 5. 自选主题演讲 bonus
- **6. 签到扣分：有事请提前与当班助教请假，有签到，无故缺席将被扣除分数。（会有值班表）**

- *ACM队参与训练无法完整参加PPCA的同学将另行通知。

项目简介

- 0. 总体规则与秩序
  - 可以交流，鼓励线上（使用聊天软件）交流
  - 可以向值班助教（如果他碰巧会你的project的话）进行提问
  - 在不离开自己的座位的情况下与周围同学小声讨论，保持机房总体秩序
  - 可以参考学长学姐在github上的repo，不推荐对着学长学姐的github项目**照抄**，严重抄袭现象将以学术诚信问题处理
  - 可以使用机房电脑或自己的电脑（机考除外）
  - 用于提高效率的外设使用，在不影响其他同学的情况下，是被允许的
  - 但请注意身边同学的感受，**不要做出类似快速而用力敲打青轴机械键盘的行为**
  - 将机房显示器用作拓展笔记本显示器时请注意使用完毕归位

03 项目简介

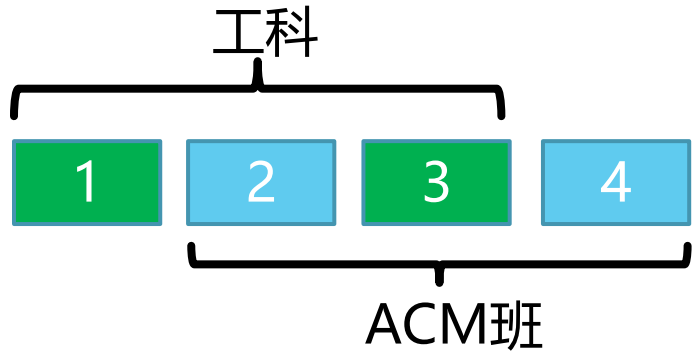
- 1. RISC-V CPU模拟器（第1周、第2周） 25
  - 第一周周一发布，第二周周日23:00 截止
  - 使用 C++模拟 RISC-V 指令集，根据乱序与否分为两个版本
  - 基本版本：五级流水，至少实现 2 位饱和计数器分支预测
  - 高级版本：托马斯洛Tomasulo乱序执行
  - 可能的Bonus：高级分支预测，数据前传（forwarding）或其他
  - Code Review（所有大作业都需要code review，影响得分，所以请养成良好的代码风格（避免hack，避免糟糕的宏定义）

  - 五级流水（基本版本）与托马斯洛（高级版本）选一实现
  - 五级流水：25，高级分支预测，数据前传 各有约1分加分
  - 托马斯洛：25+5（最高，视实现程度给分）

03 项目简介

- 2. Verilog 硬件描述语言 小作业 5
  - 不晚于第二周周一发布，第二周周日23:00 截止
  - Verilog HDL（简称 Verilog）是一种硬件描述语言，用于数字电路的系统设计。可对算法级、门级、开关级等多种抽象设计层次进行建模。
  - 学习语法，实现简单的时序逻辑与组合逻辑电路，如加法器
  - 题库：https://hdlbits.01xz.net/wiki/Problem_sets

03 项目简介



- 3. 机考 30
  - 第2周起每周四下午13:00~18:00，5小时，可提前离场，工科共3次，ACM班共5次
  - A卷为传统算法题，赛制同程序设计/数据结构课，工科前三题，ACM班后三题
  - B卷为创新综合应用题，包含一道题
  - AB二选一，最终评分只与每个人在两组试卷的答题者中的排名有关
  - 对于B卷，分数将根据答案的优劣评判（这意味着，任何格式正确的答案大都是有点分数的）
  - 如果某道题提交的答案得分达到了一条预先设定的Ground Truth的线（如果真的有的话），则此次机考这位参赛者可以不参加排名，成绩按满分计
  - 机考允许并鼓励：思考创新、寻找规律
  - 机考严禁：交流，联网查询资料（有要求除外），使用OJ上曾写过的代码，使用预先准备的模版，查看其它同学代码，把自己代码给同学看，帮助同学计算答案，其它一切不诚实行为
  - **再次强调机考学术诚信问题！！！**

项目简介

- 4. 分组项目四选一（四个项目均第三周发布，工科第四周/ACM班第六周 周五12:00 截止） 40
  - AI对战 （工科、ACM班可选）

    Minimax搜索，alpha-beta剪枝，蒙特卡洛，机器学习（ACM班）
  - Raytracer 光线追踪 （工科、ACM班可选）

    主要目标是根据RT三本书的方法实现简易的光线追踪器。增加新的材质、光照算法，GPU渲染支持（ACM班）。

    Rust语法特性。
  - DHT 分布式哈希表 （ACM班可选）

    Distributed Hash Table，chord协议、kademlia协议。
  - Raft 分布式一致性协议 （ACM班可选）
    - Consensus; Leader Election, Log Replication, Safety

- 第二周由各项目负责人进行项目具体介绍，自主报名选择项目。分数构成将于届时给出。
- 第四周/第六周下午进行结课展示。

项目简介

- 5. 自选主题演讲　bonus给分待定
  - 主动向助教报名，自定主题
  - 限制主题类主题演讲，每人20分钟左右
  - 要求 [1]： 如果是针对某个特定的领域，限制为综述
  - 要求 [2]： 如果是技术类的演讲，需要完整从背景到方法
  - 可以参考的主题有：
    - Some C++ features and design models: RAII, smart pointer, rvalue reference
    - Concurrency: mutex, lock, conditional variable, etc.
    - Common types of ISA
    - Other programming languages
  - PPCA 是一个非常好的交流想法的时机，希望有想法的同学积极报名演讲

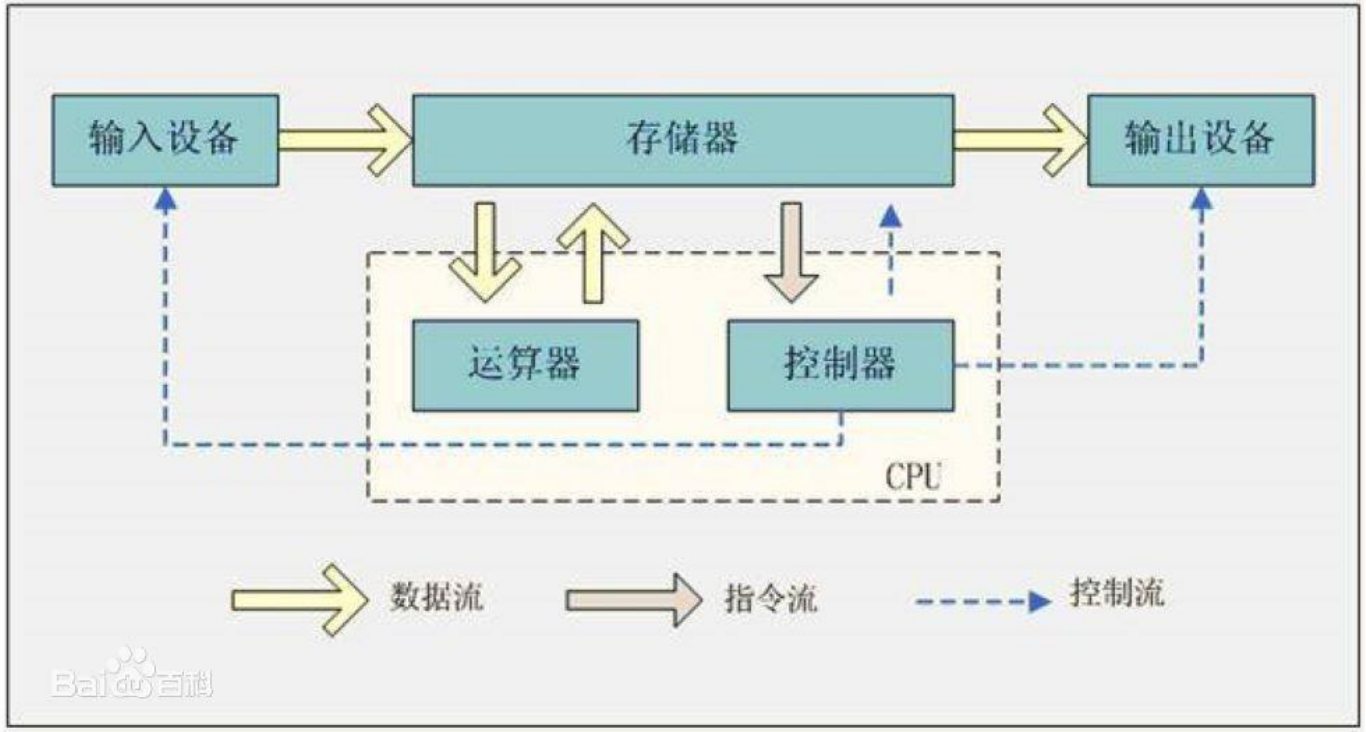
ACM：
ALWAYS CHALLENGE MIRACLES

PPCA 2021

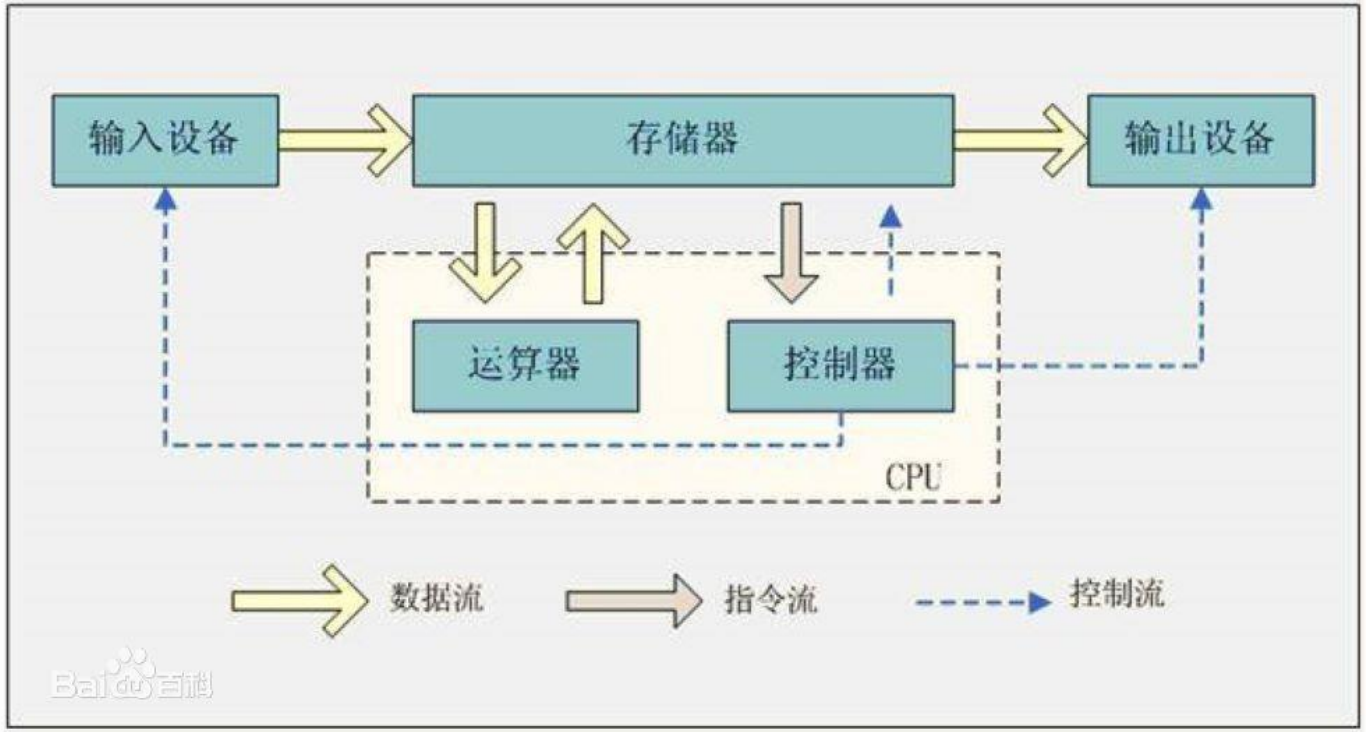
编程综合实践
Principle and Practice of Computer Algorithms
2020级ACM班 & 致远工科荣誉 2021.6.28
有问题可以随时通过QQ联系助教组!

RISC-V CPU 模拟器

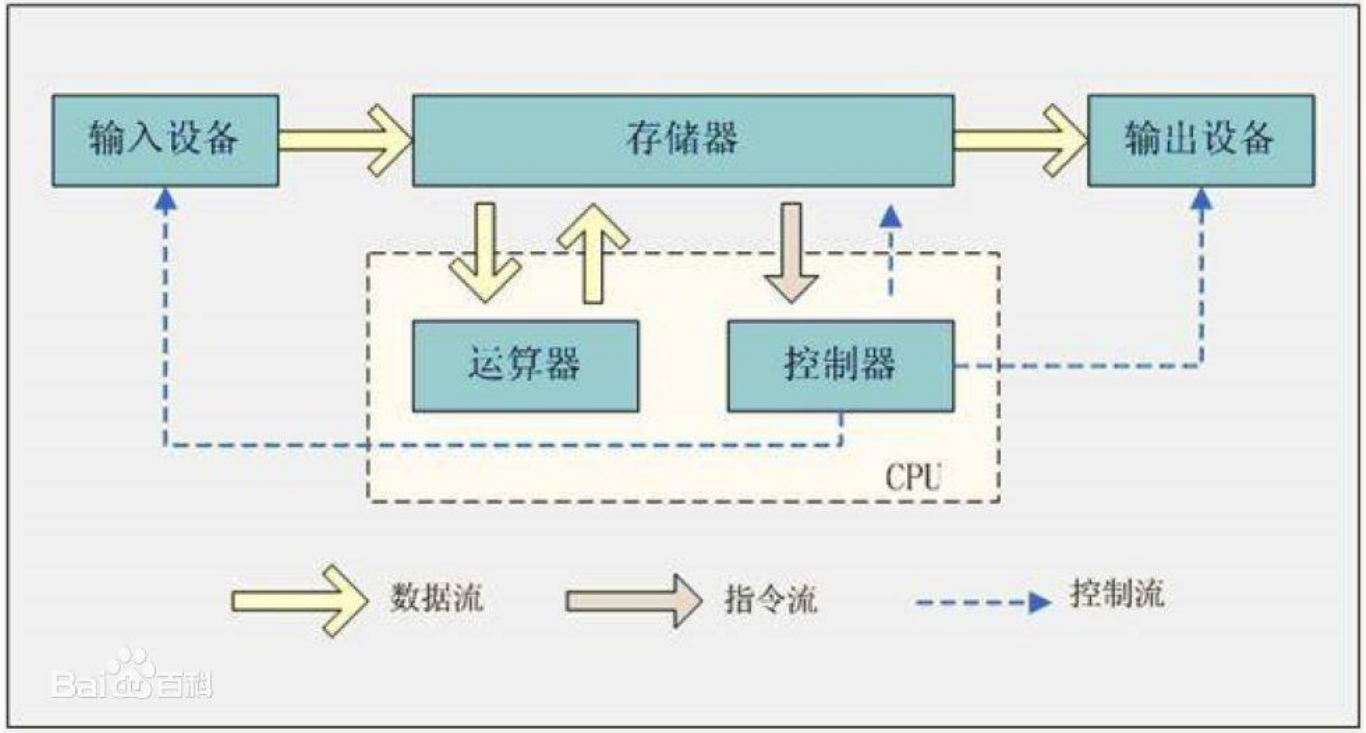
04 冯诺依曼架构



04 冯诺依曼架构



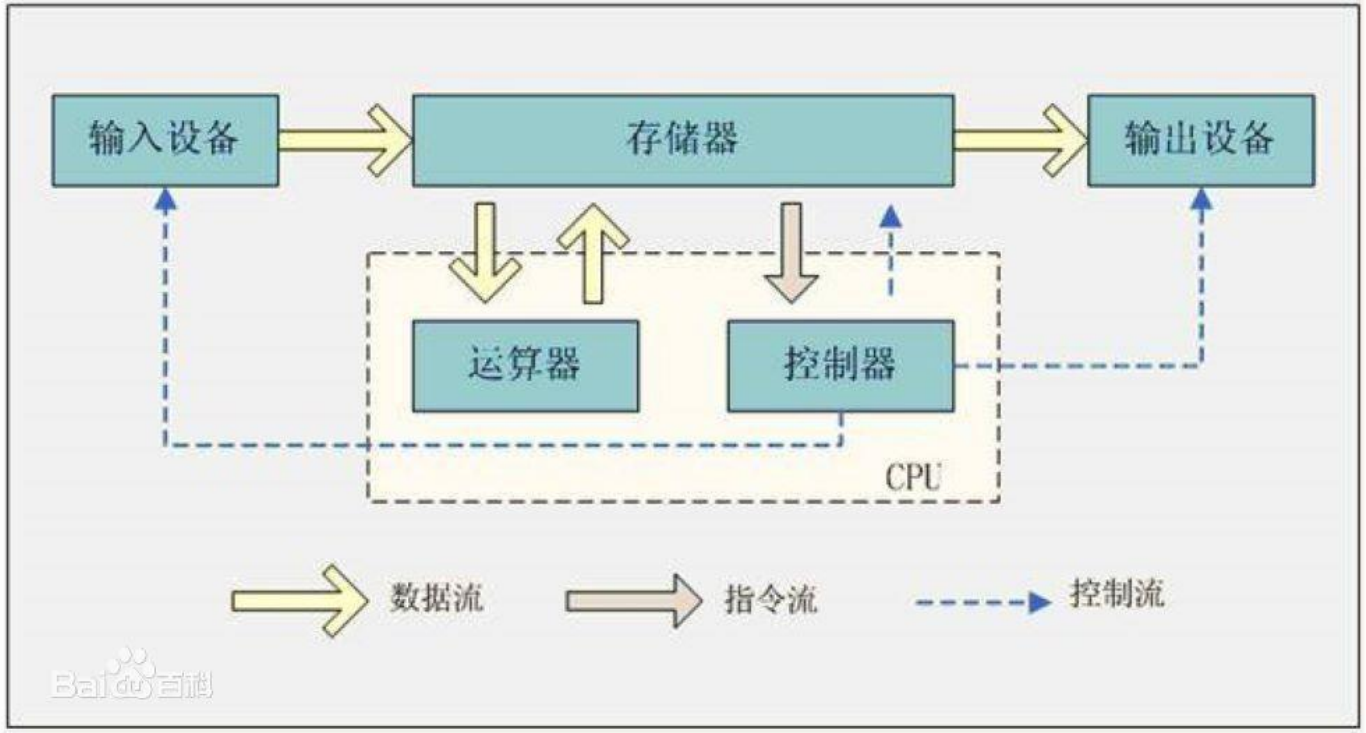
04 冯诺依曼架构



控制与运算单元

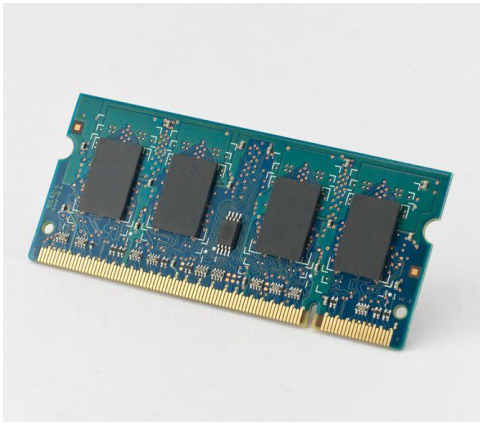
寄存器

04 冯诺依曼架构



- 控制与运算单元
- 寄存器

内存



04 概念：CPU执行指令的流程

https://www.bilibili.com/video/BV1EW411u7th 5-9

概念：CPU执行指令的流程

- 我要做什么？取指令->解码->执行
  - 开一个小数组（如unsigned reg[32]）作为寄存器；
  - 开一个大数组（如unsigned mem[500000]）作为内存；
  - 程序开始运行时，指令（是一串01串）在内存里，根据一个当前正在运行第几条指令的计数器去找内存里的第几条指令，把指令拿出来放到寄存器（Fetch Phase）
  - 根据一定的指令规范，从指令中截取出：（Decode Phase）
    - 操作码（opcode），它决定了这条指令要做什么
    - 操作对象，可能是寄存器，也可能是一个内存单元。如果某指令要对某数据做访问、修改等操作，就会用到操作对象。
  - 识别出操作码与操作对象，该运算运算，该访问访问，该修改修改（Execute Phase）

04 RISCV里的真实寄存器

- 我要做什么？取指令->解码->执行
  - 开一个小数组（如unsigned reg[32]）作为**寄存器**
  - 开一个大数组（如unsigned mem[500000]）作为
  - 程序开始运行时，指令（是一串01串）在内存里，去找内存里的第几条指令，把指令拿出来放到寄存
  - 根据一定的指令规范，从指令中截取出：（Decoc
    - 操作码（opcode），它决定了这条指令要做什
    - 操作对象，可能是寄存器，也可能是一个内存等操作，就会用到操作对象。
  - 识别出操作码与操作对象，该运算运算，该访问访

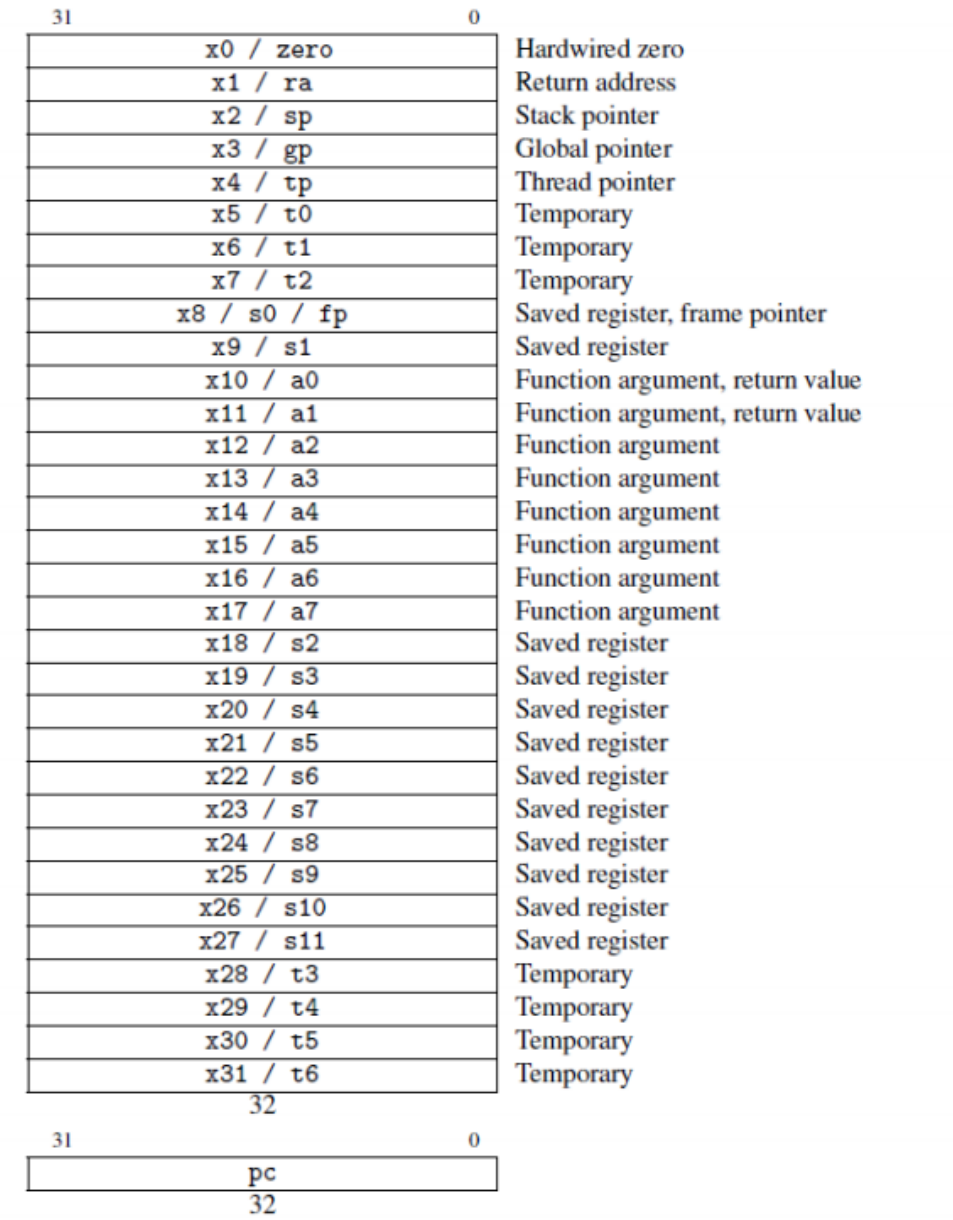


图 2.4：RV321 的寄存器。第 3 章解释了 RISC-V 调用约定，各种指针（sp，gp，tp，fp），保存寄存器（s0-s11）和临时寄存器（t0-t6）背后的基本原理（基于[Waterman and Asanovi'c 2017]的图 2.1 和表

04 RISCV里的真实指令

- 我要做什么？取指令->解码->执行
  - 开一个小数组（如unsigned reg[32]）作为寄
  - 开一个大数组（如unsigned mem[500000]）
  - 程序开始运行时，指令（是一串01串）在内存
    去找内存里的第几条指令，把指令拿出来放到
- 根据一定的**指令规范**，从指令中截取出：（D
  - 操作码（opcode），它决定了这条指令要
  - 操作对象，可能是寄存器，也可能是一个
    等操作，就会用到操作对象。
- 识别出操作码与操作对象，该运算运算，该访

imm[31:12]				rd	0110111	LUI
imm[31:12]				rd	0010111	AUIPC
imm[20|10:1|11|19:12]				rd	1101111	JAL
imm[11:0]		rs1	000	rd	1100111	JALR
imm[12|10:5]	rs2	rs1	000	imm[4:1|11]	1100011	BEQ
imm[12|10:5]	rs2	rs1	001	imm[4:1|11]	1100011	BNE
imm[12|10:5]	rs2	rs1	100	imm[4:1|11]	1100011	BLT
imm[12|10:5]	rs2	rs1	101	imm[4:1|11]	1100011	BGE
imm[12|10:5]	rs2	rs1	110	imm[4:1|11]	1100011	BLTU
imm[12|10:5]	rs2	rs1	111	imm[4:1|11]	1100011	BGEU
imm[11:0]		rs1	000	rd	0000011	LB
imm[11:0]		rs1	001	rd	0000011	LH
imm[11:0]		rs1	010	rd	0000011	LW
imm[11:0]		rs1	100	rd	0000011	LBU
imm[11:0]		rs1	101	rd	0000011	LHU
imm[11:5]	rs2	rs1	000	imm[4:0]	0100011	SB
imm[11:5]	rs2	rs1	001	imm[4:0]	0100011	SH
imm[11:5]	rs2	rs1	010	imm[4:0]	0100011	SW
imm[11:0]		rs1	000	rd	0010011	ADDI
imm[11:0]		rs1	010	rd	0010011	SLTI
imm[11:0]		rs1	011	rd	0010011	SLTIU
imm[11:0]		rs1	100	rd	0010011	XORI
imm[11:0]		rs1	110	rd	0010011	ORI
imm[11:0]		rs1	111	rd	0010011	ANDI
0000000	shamt	rs1	001	rd	0010011	SLLI
0000000	shamt	rs1	101	rd	0010011	SRLI
0100000	shamt	rs1	101	rd	0010011	SRAI
0000000	rs2	rs1	000	rd	0110011	ADD
0100000	rs2	rs1	000	rd	0110011	SUB
0000000	rs2	rs1	001	rd	0110011	SLL
0000000	rs2	rs1	010	rd	0110011	SLT
0000000	rs2	rs1	011	rd	0110011	SLTU
0000000	rs2	rs1	100	rd	0110011	XOR
0000000	rs2	rs1	101	rd	0110011	SRL
0100000	rs2	rs1	101	rd	0110011	SRA
0000000	rs2	rs1	110	rd	0110011	OR
0000000	rs2	rs1	111	rd	0110011	AND

04 RISCV里的真实指令

摘自The RISC-V Instruction Set Manual

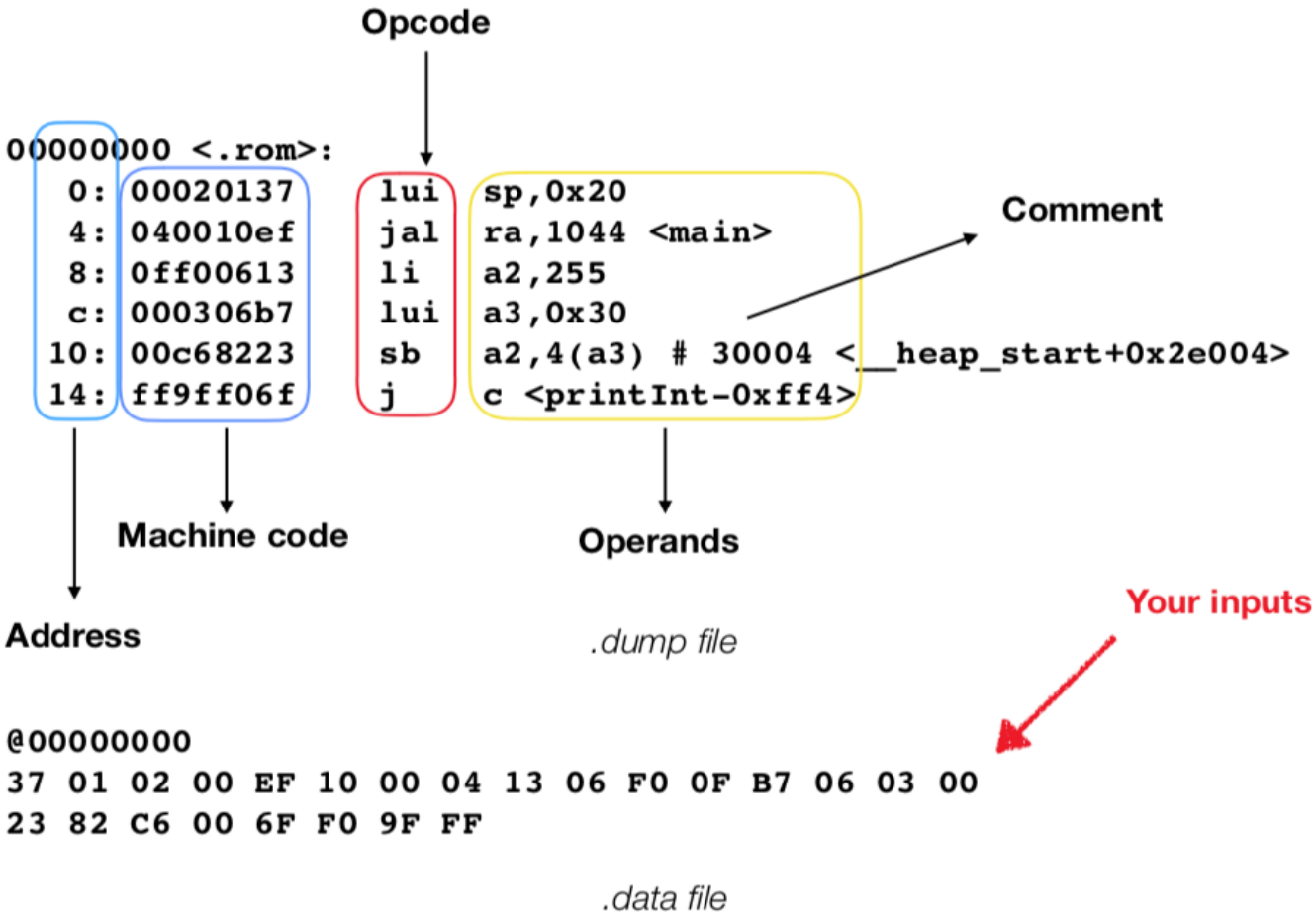
- 指令呈现方式为二进制，即使用一个32 位的01串来描述
- 每条指令的详细含义参考Chapter2
- 右图为本项目需要实现的全部指令

imm[31:12]				rd	0110111	LUI
imm[31:12]				rd	0010111	AUIPC
imm[20|10:1|11|19:12]				rd	1101111	JAL
imm[11:0]		rs1	000	rd	1100111	JALR
imm[12|10:5]	rs2	rs1	000	imm[4:1|11]	1100011	BEQ
imm[12|10:5]	rs2	rs1	001	imm[4:1|11]	1100011	BNE
imm[12|10:5]	rs2	rs1	100	imm[4:1|11]	1100011	BLT
imm[12|10:5]	rs2	rs1	101	imm[4:1|11]	1100011	BGE
imm[12|10:5]	rs2	rs1	110	imm[4:1|11]	1100011	BLTU
imm[12|10:5]	rs2	rs1	111	imm[4:1|11]	1100011	BGEU
imm[11:0]		rs1	000	rd	0000011	LB
imm[11:0]		rs1	001	rd	0000011	LH
imm[11:0]		rs1	010	rd	0000011	LW
imm[11:0]		rs1	100	rd	0000011	LBU
imm[11:0]		rs1	101	rd	0000011	LHU
imm[11:5]	rs2	rs1	000	imm[4:0]	0100011	SB
imm[11:5]	rs2	rs1	001	imm[4:0]	0100011	SH
imm[11:5]	rs2	rs1	010	imm[4:0]	0100011	SW
imm[11:0]		rs1	000	rd	0010011	ADDI
imm[11:0]		rs1	010	rd	0010011	SLTI
imm[11:0]		rs1	011	rd	0010011	SLTIU
imm[11:0]		rs1	100	rd	0010011	XORI
imm[11:0]		rs1	110	rd	0010011	ORI
imm[11:0]		rs1	111	rd	0010011	ANDI
0000000	shamt	rs1	001	rd	0010011	SLLI
0000000	shamt	rs1	101	rd	0010011	SRLI
0100000	shamt	rs1	101	rd	0010011	SRAI
0000000	rs2	rs1	000	rd	0110011	ADD
0100000	rs2	rs1	000	rd	0110011	SUB
0000000	rs2	rs1	001	rd	0110011	SLL
0000000	rs2	rs1	010	rd	0110011	SLT
0000000	rs2	rs1	011	rd	0110011	SLTU
0000000	rs2	rs1	100	rd	0110011	XOR
0000000	rs2	rs1	101	rd	0110011	SRL
0100000	rs2	rs1	101	rd	0110011	SRA
0000000	rs2	rs1	110	rd	0110011	OR
0000000	rs2	rs1	111	rd	0110011	AND

RISCV里的真实指令

- RISC-V Assembly
- 汇编语言 (assembly language) 的一种
- 是基于计算机基础结构的基础语言
- 每条指令语句都会是简单的，仅涉及三个元素的操作
- Reference: Chapter 2, The RISC-V Instruction Set Manual, VOL1（参考2.1-2.7即可）
- 使用RV32I基础整数指令集 (RV32I Base Integer ISA)

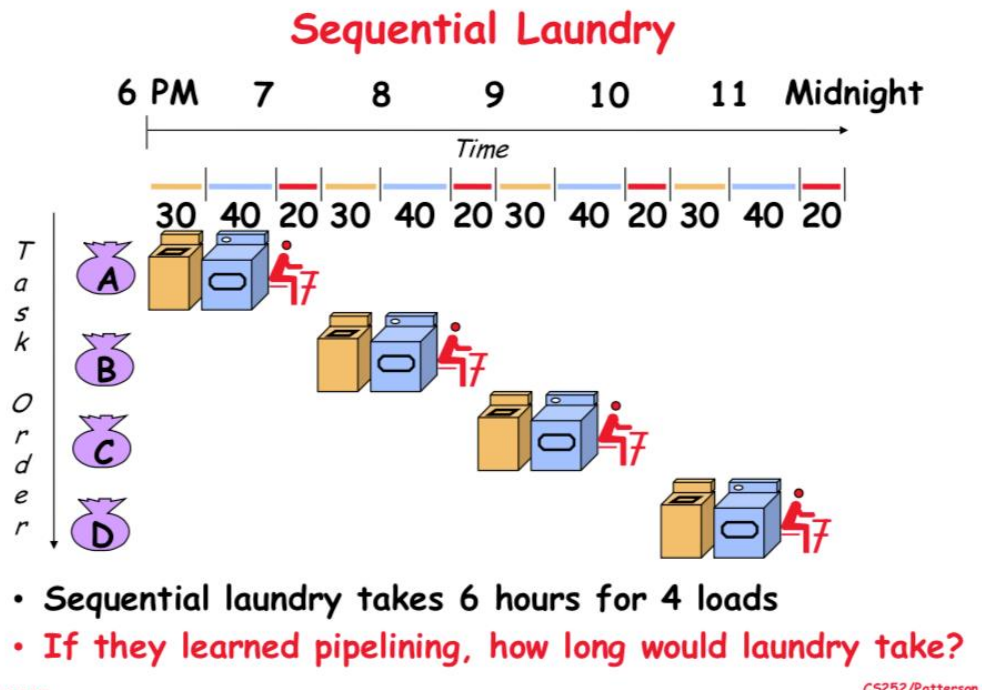
04 RISCV里的真实指令

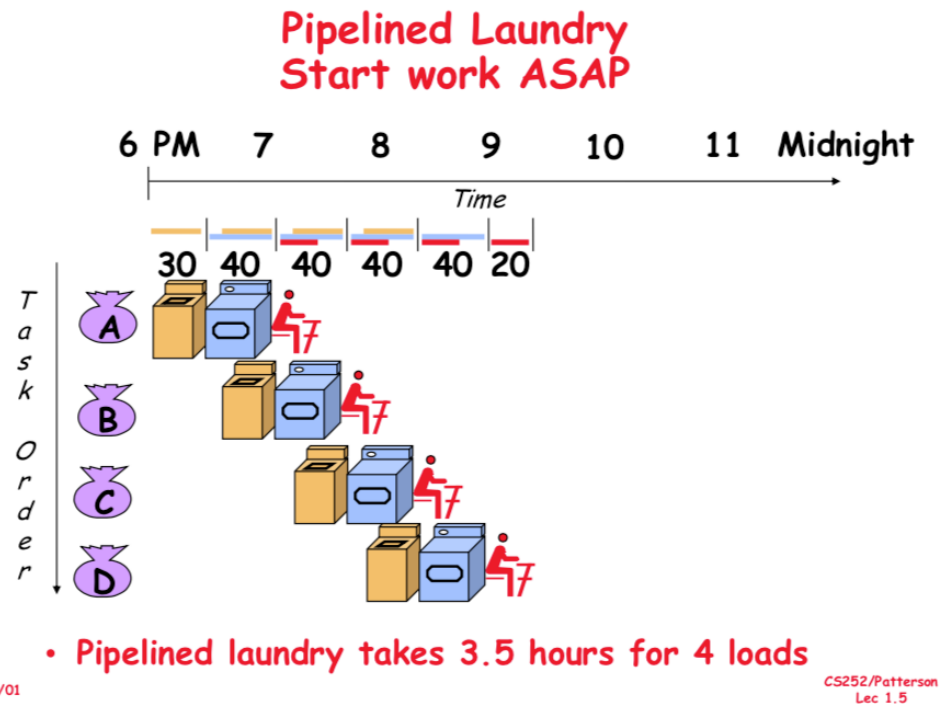


04 Pipeline：五级流水

Reference: CAAQA, John L. Hennessy & David A. Patterson

参考：《计算机体系结构：量化研究方法》





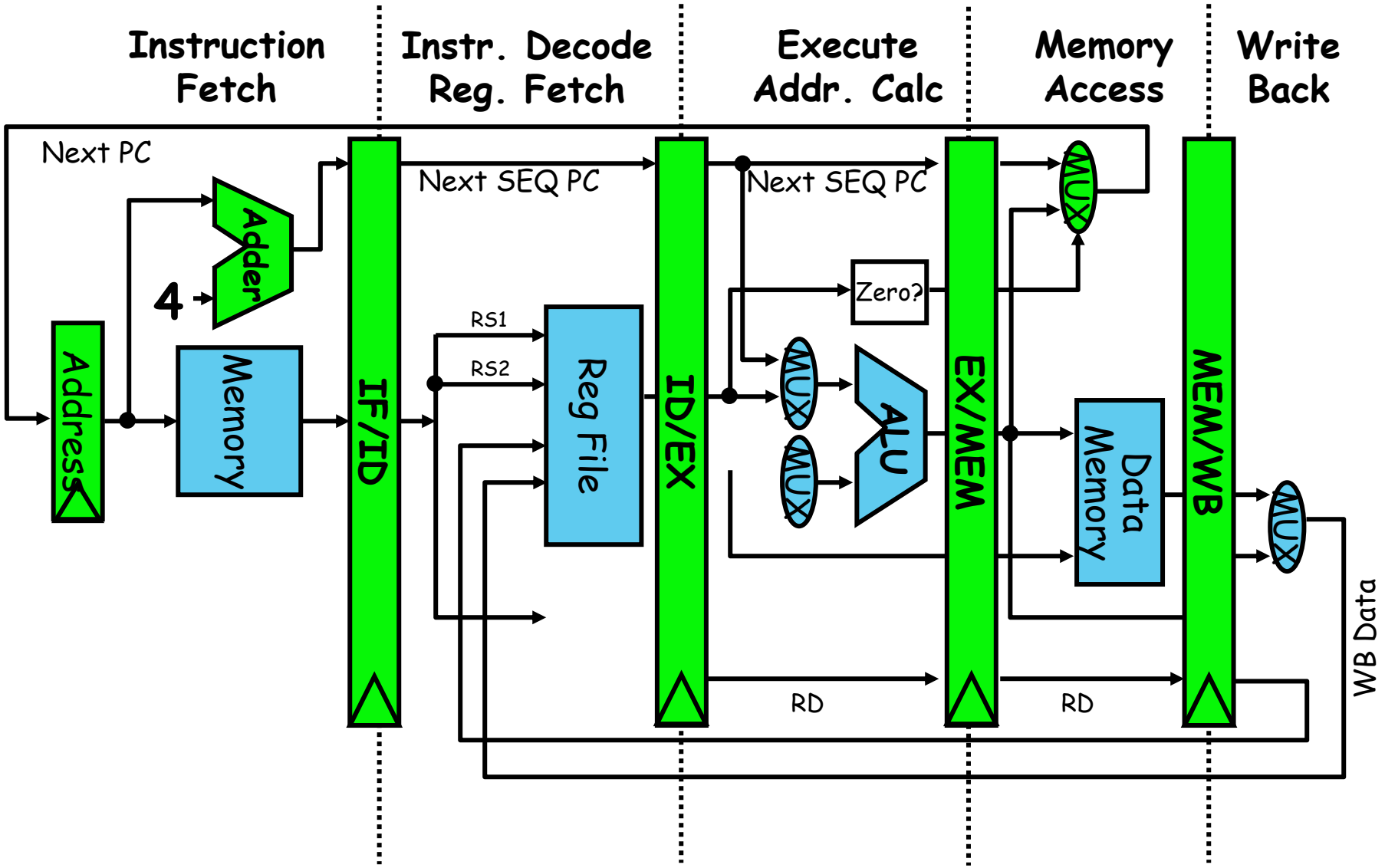
Pipeline：五级流水

五级流水 5-Stage Pipeline

CPU 6%

- Instruction Fetch (IF)：根据PC寄存器访问内存得到指令
- Instruction Decode (ID)：根据指令类型找到作为右值的寄存器并读值、并且解析立即数的值
- Execute (EX)：对解析好的值按指令要求进行计算
- Memory Access (MEM)：根据计算出的地址从内存读出数据值，或将已准备好的数据值写入内存
- Write Back (WB)：完成对于左值寄存器的赋值，即写回寄存器

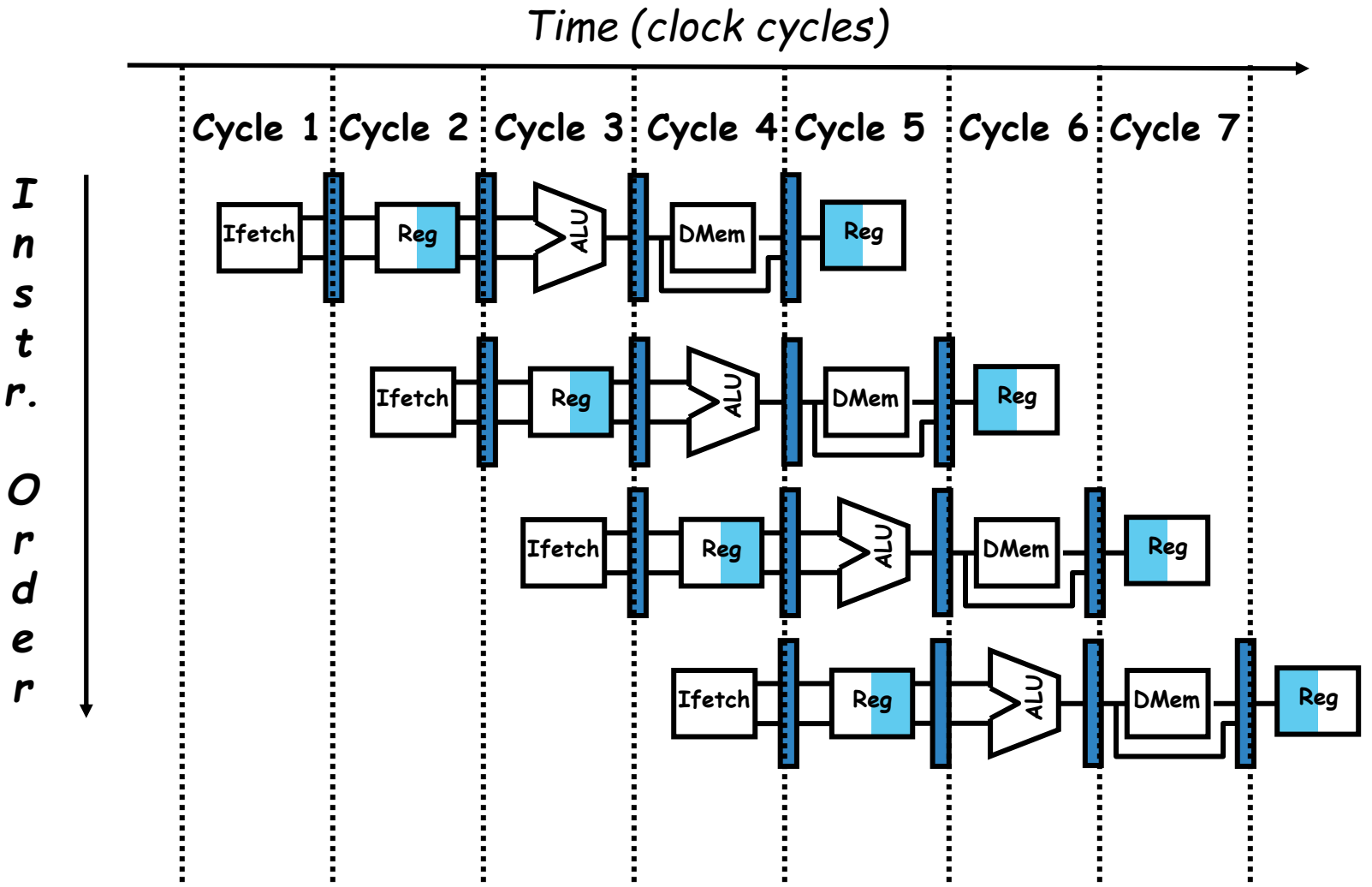
04 Pipeline：五级流水



04 Pipeline：五级流水

Visualizing Pipelining

Figure 3.3, Page 133 , CA:AQA 2e



Pipeline：五级流水

Hazards

- Structural hazard（模拟时不会出现这种hazard）
- Data hazard
- Control hazard
- 最简单的解决方案即等待闲置，暂停流水

Pipeline：五级流水

分支预测

- 遇到跳转分支指令时的预测，最简单的预测方案为：总是跳转，或总是不跳转（但这种预测方案不符合本次作业分支预测的最低要求）
- 最低要求：实现一个2位饱和计数器分支预测，当然也可以实现更高级的分支预测
- 支持留出接口统计分支预测成功率并展示之

Pipeline：五级流水

Simulation

- 给出的数据中含有.c, .dump和.data文件
- Input (stdin)：.data 文件
- Output (stdout)：模拟器运行结果
- .c文件和.dump文件可供参考

04 RISCV里的真实指令

- 执行流程：
  - 从标准输入读入机器指令
  - 从内存00000000处开始取指令执行，每次连续取4个2位十六进制数，组成一条指令（如取到“37 17 00 00”，拼成32位指令“00001737”）
  - load/store指令内存访问部分（第四级）请用**三个周期**模拟
  - 执行到指令0ff00513（li a0,255）时，向标准输出 输出程序的返回值（一个0-255的非负整数），结束模拟。注意：
    - 程序的返回值存在a0寄存器里，但是寄存器是32位的，返回值是8位的，所以你应该输出a0的后八位。例如，你的a0寄存器是int数组reg中的reg[10]，你**应该输出**的是((unsigned int)reg[10]) & 255u。

RISCV里的真实指令

建议流程：

熟悉RISCV指令集，弄懂机器指令的执行流程

写一个不带流水的简易模拟器

加上五级流水

写写bonus，欢迎实现各种个性化的功能