



★ コンピュータアーキテクチャ

坂井 修一



计算机组成与设计 MIPS版

David A. Patterson

计算机系统层次结构

硬件+软件

冯诺依曼结构

① 存储程序 ② 输入设备、输出设备、运算器、存储器、控制器

③ 以运算器为中心 ④ 指令与数据同等形式存储, (二进制). 指令由操作码+地址码

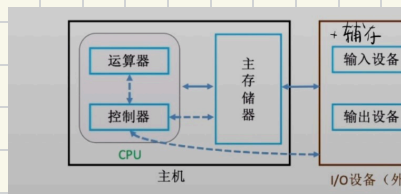
指令集

RISC, CISC

MIPS, ARM

现代计算机结构: 以存储器为中心

辅存视作 I/O 设备



MIPS 汇编语言

算术逻辑、读写、跳转

Register: 在CPU内存储数据、数量少、速度快

ADD R1, R2, R3: $R1 = R2 + R3$

SUB R1, R2, R3: $R1 = R2 - R3$

ADDI R1, R2, Imm: $R1 = R2 + Imm$ ← 立即数, 不是寄存器

浮点数

规格化 见PPT(F)

$$M \times r^E, \quad M \text{ 小数点前只有一位且非 } 0$$

左规: 尾数数值左移1位, 阶码-1

右规: 尾数数值右移1位, 阶码+1

IEEE 754

	Sign	Exponent	Mantissa	符号, 阶码, 尾数	偏置值	阶码偏置
单精度 32bit:	1	8	23		127	
双精度 64bit:	1	11	52		1023	

eg (0.75)₁₀ = (0.11)₂ $\xrightarrow[\text{左规}]{\text{规格化}}$ 1.1×2^{-1} $\xrightarrow[\text{第一位必为1, 不用存储}]{\text{隐藏最高位}}$ 0.1×2^{-1} $\xrightarrow[\text{偏置阶码}]{\text{计算阶码}}$ $-1 + 127 = 126$ $\xrightarrow[\text{+1}]{\text{加符号}}$ $0 \mid 01111110 \mid 10 \dots 0$

IEEE754 $\rightarrow$ 真值, $N = (-1)^S \times 2^{E - \text{offset}(n)}$ $\times (1 + M)$

加减法 ① 计算阶码真值 ② 对阶 (小阶对大阶), 移位 ③ 0舍1入 (舍弃的首位为1, 进位) ④ 尾数加减

流水线

单周期处理器: 一个时钟周期一个任务

多周期处理器模型

细分 Stages

- ① IF, Instruction Fetch, 取指令 用PC取出
- ② ID, Instruction Decode 编译指令, 解析指令 (通过密码器), 也取数据
- ③ EXE, Execution 执行, 运算
- ④ MEM, Memory Access 地址访问, 取地址中对应的值
- ⑤ WB, Write Back 写回寄存器

lw	sw	运算	beq
✓	✓	✓	✓
✓	✓	✓	✓
✓	✓	✓	✓
✓	✓		
✓	✓	✓	

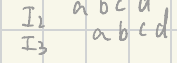
一条指令多个周期, 一个周期执行一个阶段. 按最慢的阶段划分为周期

周期 $\downarrow$, 效率 $\uparrow$

串行: abc dabc d...

并行: 流水线结构 时间重叠, 并行处理.

物理结构不重复占用, 多条指令周期+1, CPI=1

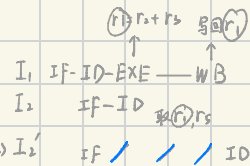


EXE + MEM

也有四周期流水线: F D E W

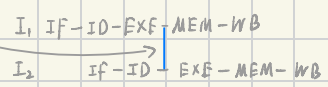
冒险

① 数据冒险: 下一条指令会用到当前指令的结果



Workround: 添加气泡 (stall): 暂停与数据有关的指令

前推/转发 (bypassing)

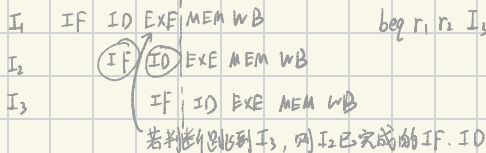


编译优化 (scheduling)

② 结构冒险: 多条指令争用同一资源, 缺乏硬件支持。

③ 控制冒险: (分支指令一定出现)

workaround:



添加 stall

分支预测: 预测对: $\checkmark$ 预测错: 需添加 stall

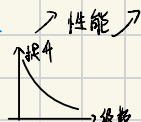
数据依赖性

RAW 先写后读 会有数据冒险

WAR

WAW

流水级数



周期变少

分支指令 (分支冒险浪费多)
设计复杂

作业

- (3) 以下の同じ命令セットアーキテクチャの三つのプロセッサにおいて、実行命令数が 1,000,000 で、そのうち 20% が条件分岐命令であるプログラムを実行した場合の計算時間を求めよ。条件分岐命令の 30% で条件が成立するものとする。
- (a) クロック・サイクル時間が 2ns の単一サイクル方式のプロセッサ。
- (b) クロック・サイクル時間が 500ps の 5 段パイプライン方式のプロセッサ。ただし、条件分岐命令で条件が成立した場合、1 サイクルストールする。
- (c) クロック・サイクル時間が 300ps の 10 段パイプライン方式のプロセッサ。ただし、条件分岐命令で条件が成立した場合、2 サイクルストールする。
- (3) Show the execution time of a program of 1,000,000 executed instructions, 20% of which are conditional branch instructions, on the following three processors with the same instruction set architecture. The condition is taken for 30% of the conditional branch instructions.
- (a) Single-cycle implementation with clock cycle time of 2 ns.
- (b) 5-stage pipelined implementation with clock cycle time of 500 ps which stalls one cycle for a conditional branch instruction when the condition is taken.
- (c) 10-stage pipelined implementation with clock cycle time of 300 ps which stalls two cycles for a conditional branch instruction when the condition is taken.

① I₁ I₂ ② I₃ I₆

③ I₄ I₅

I₁ F D R E M B

I₂ F D R / E M B

I₃ F D R / E M B

I₄ F D R / E M B

I₅ F D R / E M B

I₆ F D / R E M B

single-issue

以下の問いに答えよ。

- (1) 下記のアセンブリコードは、メモリに格納されている配列の要素を順次読み出しながら要素の総和を求め、指定されたメモリアドレスに結果を格納するプログラムの計算部分である。コード中、r1~r6 はラベルであり、r1~r5 はレジスタを表す。レジスタ r1, r2, r5 の値は、それぞれ配列の要素数、配列の先頭アドレス、結果を格納するアドレスに初期化されているものとする。下記コードの四角で囲まれた空欄 a から e について、レジスタ番号または即値を入れて完成せよ。

```
11: LD    r3    0(r2)
12: ADD  r4    r3
13: ADDI r2    r2    8
14: ADDI r1    [a] [b]
15: BNZ  [c]    I1:
16: ST   [d]    0(r5)
```

- (2) 以下に示す命令対 (i), (ii) について、該当する依存関係を下の a, b, c の選択肢より選べ。

(i) 命令 I1 と命令 I2

(ii) 命令 I1 と命令 I3

- a. Read After Write (フロー依存)
b. Write After Read (逆依存)
c. Write After Write (出力依存)

- (3) ステータス構成と各ステージの処理時間が以下の表で与えられるようなパイプラインプロセッサを考へる。このプロセッサの動作周波数を示せ。各ステージは 1 サイクルの間に完了する。またクロックスキュー対策などによるアービタリズに表中の数値に含まれているものとしてよい。なお、[ps] は 10^{-12} 秒である。

フェッチ	250 [ps]
デコード	100 [ps]
レジスタリード	150 [ps]
エクゼキュート	100 [ps]
メモリアクセス	300 [ps]
レジスタライト	150 [ps]

- (4) このパイプラインプロセッサで (1) のコードを実行する。バブルの発生が予想される命令が連続存在するが、全てのそのような命令と、該当するハザードを示せ。プロセッサはシングルイシュー・カウインオーダーとす。オペランドフォワードイング回路を仮定してもよい。

Answer the following questions.

- (1) The following assembly code is a calculation part of a program which sequentially reads the elements of an array stored in the memory, obtains a sum of all elements, and stores the result in a designated memory address. In this code, r1~r5 indicate registers, and r1 to r5 indicate registers. Assume that the values of register r1, r2, and r5 are initialized to the number of elements in the array, the beginning address of the array, and the address for storing the result, respectively. Fill in the each boxed blank a to e using an adequate register number or an immediate value.

```
11: LD    r3    0(r2)
12: ADD  r4    r3
13: ADDI r2    r2    8
14: ADDI r1    [a] [b]
15: BNZ  [c]    I1:
16: ST   [d]    0(r5)
```

- (2) Answer the relevant data dependency for the following instruction pairs (i) and (ii), respectively. Choose the correct answer from the following a, b, and c.

(i) instruction I1 and instruction I2 R A W a

(ii) instruction I1 and instruction I3 W A R b

- a. Read After Write (Flow dependency)
b. Write After Read (Anti-dependency)
c. Write After Write (Output dependency)

- (3) Consider a pipelined processor whose stage organization and processing times are given in the following table. Answer the operation frequency of this processor. Each stage is completed within a cycle. You can assume that margins such as for mitigating clock skew are already involved in the values in the table. Note that [ps] represents 10^{-12} seconds.

$T = 1050 \text{ ps}$

4 - 4 T, + 3 300 ps

$T = 300 \text{ ps} = 300 \times 10^{-12}$

$f = \frac{1}{T} = \frac{1}{300 \times 10^{-12}}$

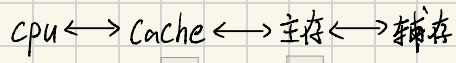
- (4) Consider that the code in (1) is executed with this pipelined processor. Here several instructions are expected to generate pipeline hazards. Answer all of such instructions and the name of the relevant hazard type for each. Assume that the processor is single issue and in-order. You can also assume the operand forwarding.

後序、後読み後不能早比較指令早出来

不能在同周期执行同一命令

存储器

主存、辅存、Cache (与CPU速度匹配, 价格高)



MHz (CPU: GHz) → 不匹配: 冯诺依曼瓶颈, Cache用以缓解

磁带: 顺序存取, 半导体: 随机存取, 存储时间与存储单元无关

主存 DRAM → 电容 触发器 ← SRAM Cache

读 破坏性读出 查看高低电平

写 充放电 改变状态

刷新: DRAM电容需要不断刷新以防止信息丢失。(电荷仅维持1~2ms) 读写一次 0.5μs 需要硬件支持

刷新周期一般为2ms.

① 集中刷新 每2ms刷新一次, 形成不能R/W的dead zone.

② 分散刷新 对每一个电容独立刷新周期.

③ 异步刷新 统一刷新, 但周期小于2ms, 死区较少.

RAM: 随机存储器 断电丢失数据 主存

ROM: 只读存储器 断电数据保留 辅存(可写) MROM, PROM, EPROM, EEPROM...

Cache

(block)

与主存一样划分为等大的块, 块数远小于主存. 仅保存主存中最活跃的block.

局部性原理 局所性

1. 时间局部性, Temporal Locality, 如果某数据被访问, 则不久未来可能再次访问

2. 空间局部性, Spatial Locality, 则与之相邻的数据可能被访问

命中/未命中 Hit/Miss Hit Rate = $N_c / (N_c + N_m) \times 100\%$

映射关系 (Cache与主存的mapping)

全相联

直接

哪都行

一一对应, 多对少

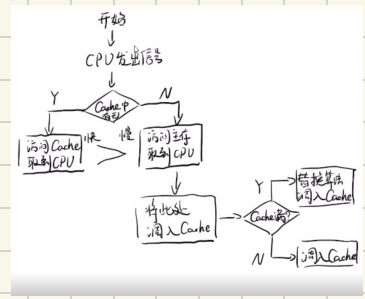
Cache 块号 = 主存块号 % Cache块数

Pros 灵活, 利用率高

不灵活, 利用率低, 冲突概率高

Cons 慢

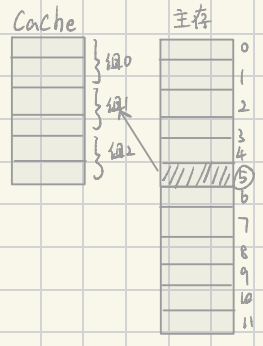
快



组相联映射 n块为一组 → n路组相联, 组间直接映射, 组内全相联映射

↑ 标记位: 在Cache块中标记来源主存块 初始为0, 存tag

先看 → 有效位: 0为空, 1为非空 解决'0'是否为空的问题



① Cache 组号 = 主存块号 % 组数
② 组内用全相联自由选取

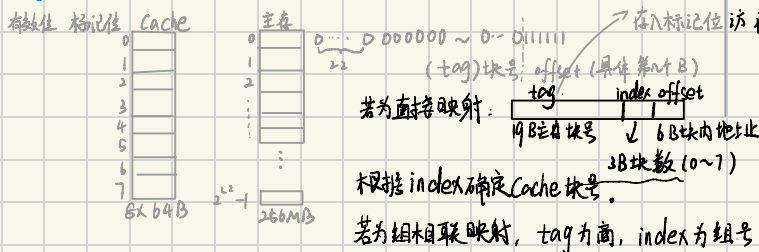
$5 \% 3 = 1$

编址方式

按Byte编址

按字(word=4B)编址

e.g. 主存 256MB, 按字节编址, 8块 Cache, 64B/块, 全相联映射。



- 存入标记位 访存:
- ① 对比标记位与主存块号
 - ② 若有, 有效位为 1, hit
 - ③ 未命中/有效位为 0, miss, 从主存读
 - ④ 进入 Cache.
- 主存块号 $\div$ 组数 = tag ... index

替换方式

组相联: 组内选择

RAND: 随机

FIFO: 先进先出

LRU: 近期最少使用 Least Recently 时间

LFU: 最不经使用 Least frequency 频率

MISS 种类

- ① Compulsory miss/cold miss: 第一次使用
- ② Capacity miss: 满了新块无法存入
- ③ Conflict miss: 目标位置被占用, 替换

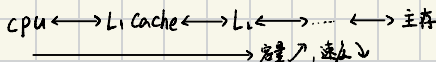
Cache 相联性 Associativity

N 路组相联映射 $\equiv N$ 块 Cache 直接映射

分离 Cache 架构.

指令 Cache, 数据 Cache 分开存放 (防止 IPU 占用, 使用各自局部性原理)

多级 Cache 架构



Cache 写策略

Write hit:

- ① write through: 同时改写 Cache 和主存
- ② write back: 先改写 Cache, 被读出时才写回到主存。Cache 添一个脏位, 记录是否与主存不一致。

Write miss:

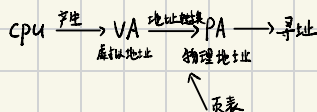
- ① write-allocate: 写入主存, 调入 Cache
- ② not-write-allocate

虚拟存储器 主存+辅存, 按需调用辅存进入内存

页式: 等大小分割: 页内碎片浪费

段式: 逻辑分割成不等大小: 段外碎片浪费

段页式



VA: 虚页号 + offset (页内地址). PA: 页表 (有效位 + VA-PA 映射)

1	PA of VA
0	DA

有效位为 1

page fault 缺页 主存中找不到VA对应的PA → 由OS控制, 根据页表中的 Disk Address 去辅存查。

→ 放入主存(放入/替换)

TLB 快表 Cache组成的页表, 主存中的页表称为慢表。相当于页表的缓存。

并行处理

流水线

指令级并行 ILP: 指令在流水线中可以重叠起来运行。

单核: 时间重叠; 多核: 并行处理, 硬件支持



超长指令集架构 VLIW 多指令并行 同时执行

封装在一起的指令无依赖性: 避免数据冒险

CPU、硬件支持 避免结构冒险

依赖性由 compiler 决定

cons: 不灵活, 静态

超标量体系结构 superscalar 动态判断是否封装

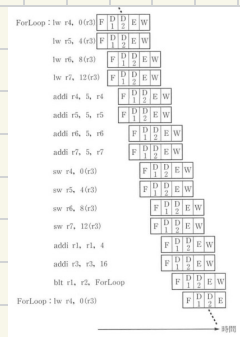
并行处理冒险

结构冒险: ALU等不够

数据冒险: RAW, WAR, WAW

分支冒险

循环展开 loop unrolling: 避免添加 stall



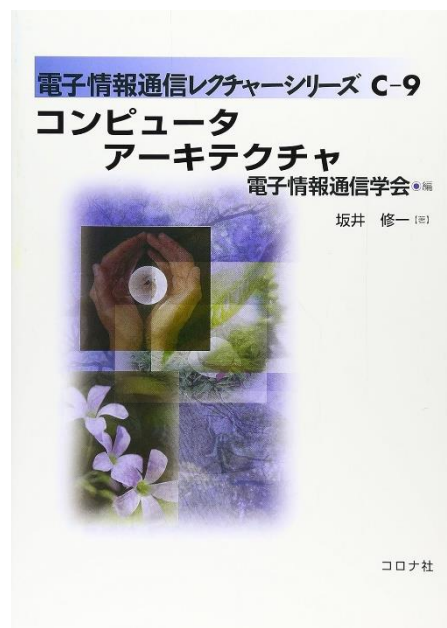
计算机系统 & 汇编语言概述

计算机组成原理 第一讲

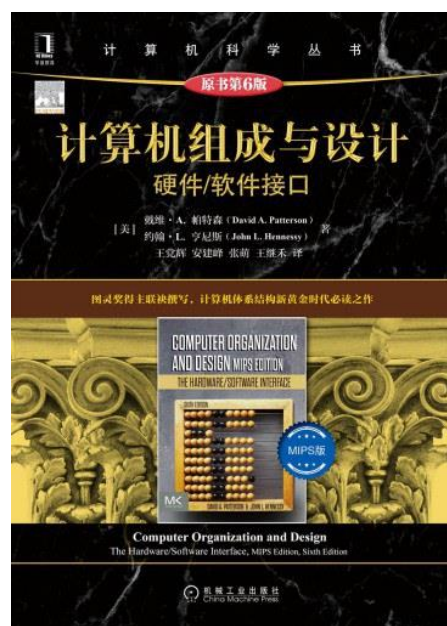


尚岸塾

参考教材



コンピュータアーキテクチャ 坂井 修一



计算机组成与设计 MIPS版 David A. Patterson

本讲主要内容

1. 计算机系统层次结构
2. MIPS格式下的汇编语言
3. 计算机性能指标

计算机系统层次结构

计算机系统层次结构

硬件系统和软件系统共同构成了一个完整的计算机系统。

计算机系统 = 硬件 + 软件

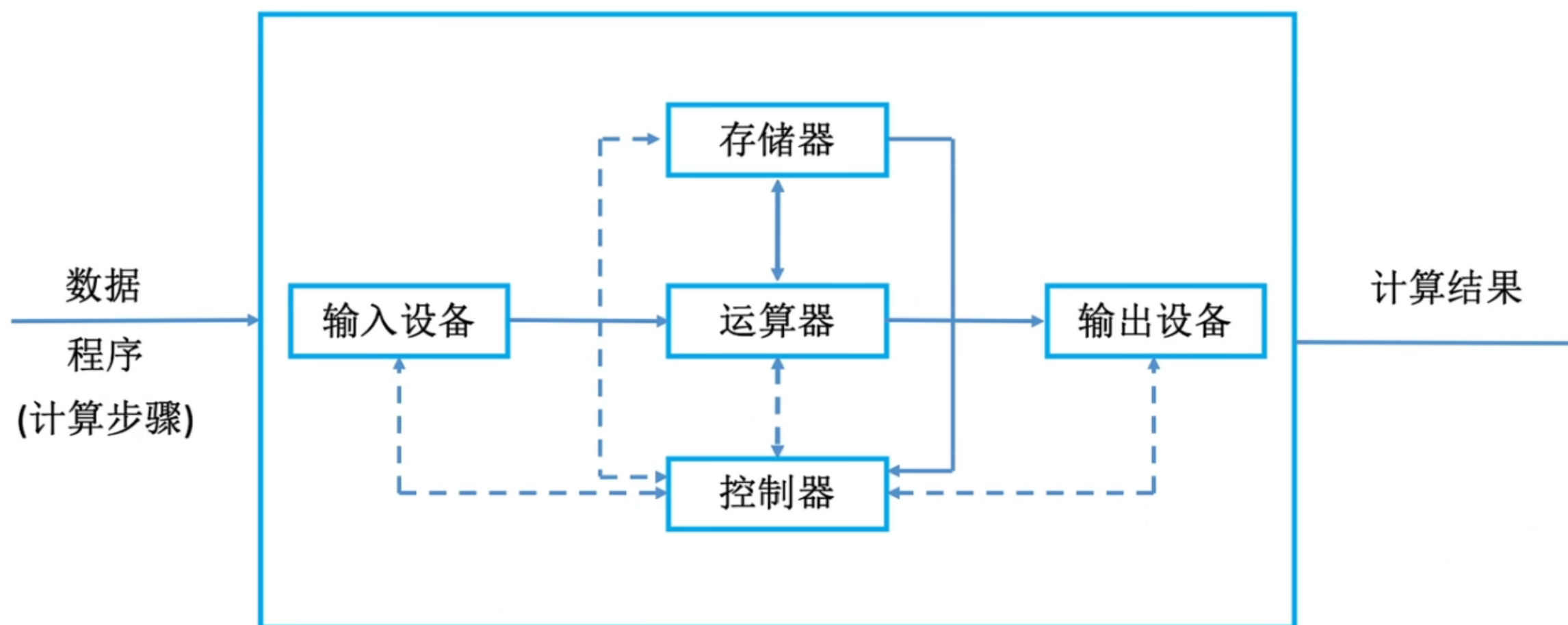
计算机系统层次结构

冯诺依曼结构：

1. 采用“存储程序”的工作方式；
2. 计算机硬件系统由运算器、存储器、控制器、输入设备和输出设备5大部件组成；
3. 指令和数据以同等地位存储在存储器中，形式上没有区别，但计算机应能区分它们；
4. 指令和数据均用二进制代码表示。指令由操作码和地址码组成，操作码指出操作的类型，地址码指出操作数的地址；
5. 以运算器为中心。

计算机系统层次结构

冯诺依曼结构：



计算机系统层次结构

指令集：

存放一切需要调用的指令的集合

高级语言

汇编语言

机器语言

RISC (reduced instruction set computer)

CISC (complex instruction set computer)

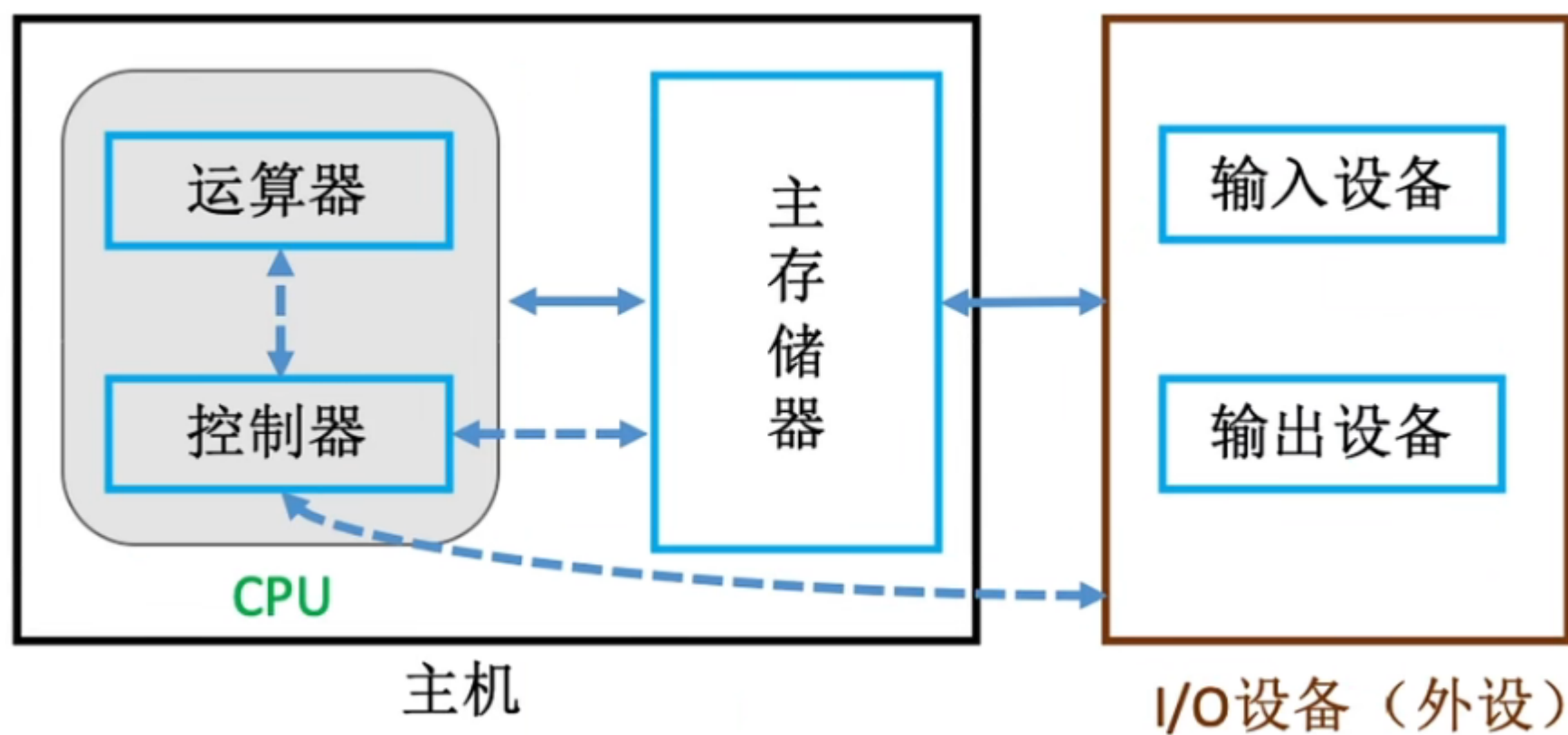
MIPS

计算机系统层次结构

现代计算机结构：

计算机系统层次结构

现代计算机结构：



MIPS格式下的汇编语言

MIPS格式下的汇编语言

指令长度

op: 操作码

rs, rt: 源操作寄存器] 地址码

rd: 目标

shamt: 位移量

funct: 功能码

表 (a) 命令形式と命令セット (表内の数字はすべて 10 進数).

Table (a) Instruction formats and instruction set (all numbers in this table are decimal).

命令 Inst- ruction	形式 Format	フィールド Field						アセンブリ・コード Assembly code 動作 Operation
	R	op	rs	rt	rd	shamt	funct	
	width	6b	5b	5b	5b	5b	6b	
	I	op	rs	rt	Immediate			
	width	6b	5b	5b	16b			
add	R	0	rs	rt	rd	0	32	add rd, rs, rt addition: $R[rd] = R[rs] + R[rt]$ 値操作
sub	R	0	rs	rt	rd	0	34	sub rd, rs, rt subtraction: $R[rd] = R[rs] - R[rt]$
and	R	0	rs	rt	rd	0	36	and rd, rs, rt logical AND: $R[rd] = R[rs] \& R[rt]$
or	R	0	rs	rt	rd	0	37	or rd, rs, rt logical OR: $R[rd] = R[rs] R[rt]$
nor	R	0	rs	rt	rd	0	39	nor rd, rs, rt logical NOR: $R[rd] = \sim(R[rs] R[rt])$
addi	I	8	rs	rt	Imm			addi rt, rs, Imm add immediate: $R[rt] = R[rs] + \text{SignExtImm}$
andi	I	12	rs	rt	Imm			andi rt, rs, Imm AND immediate: $R[rt] = R[rs] \& \text{ZeroExtImm}$
ori	I	13	rs	rt	Imm			ori rt, rs, Imm OR immediate: $R[rt] = R[rs] \text{ZeroExtImm}$
lw	I	35	rs	rt	Imm			lw rt, Imm(rs) load word: $R[rt] = M[R[rs] + \text{SignExtImm}]$
sw	I	43	rs	rt	Imm			sw rt, Imm(rs) store word: $M[R[rs] + \text{SignExtImm}] = R[rt]$
sll	R	0	0	rt	rd	shamt	0	sll rd, rt, shamt logical shift left: $R[rd] = R[rt] \ll \text{shamt}$
srl	R	0	0	rt	rd	shamt	2	srl rd, rt, shamt logical shift right: $R[rd] = R[rt] \gg \text{shamt}$
sra	R	0	0	rt	rd	shamt	3	sra rd, rt, shamt arithmetic shift right: $R[rd] = R[rt] \gg \text{shamt}$

读写指令 { load word
store word

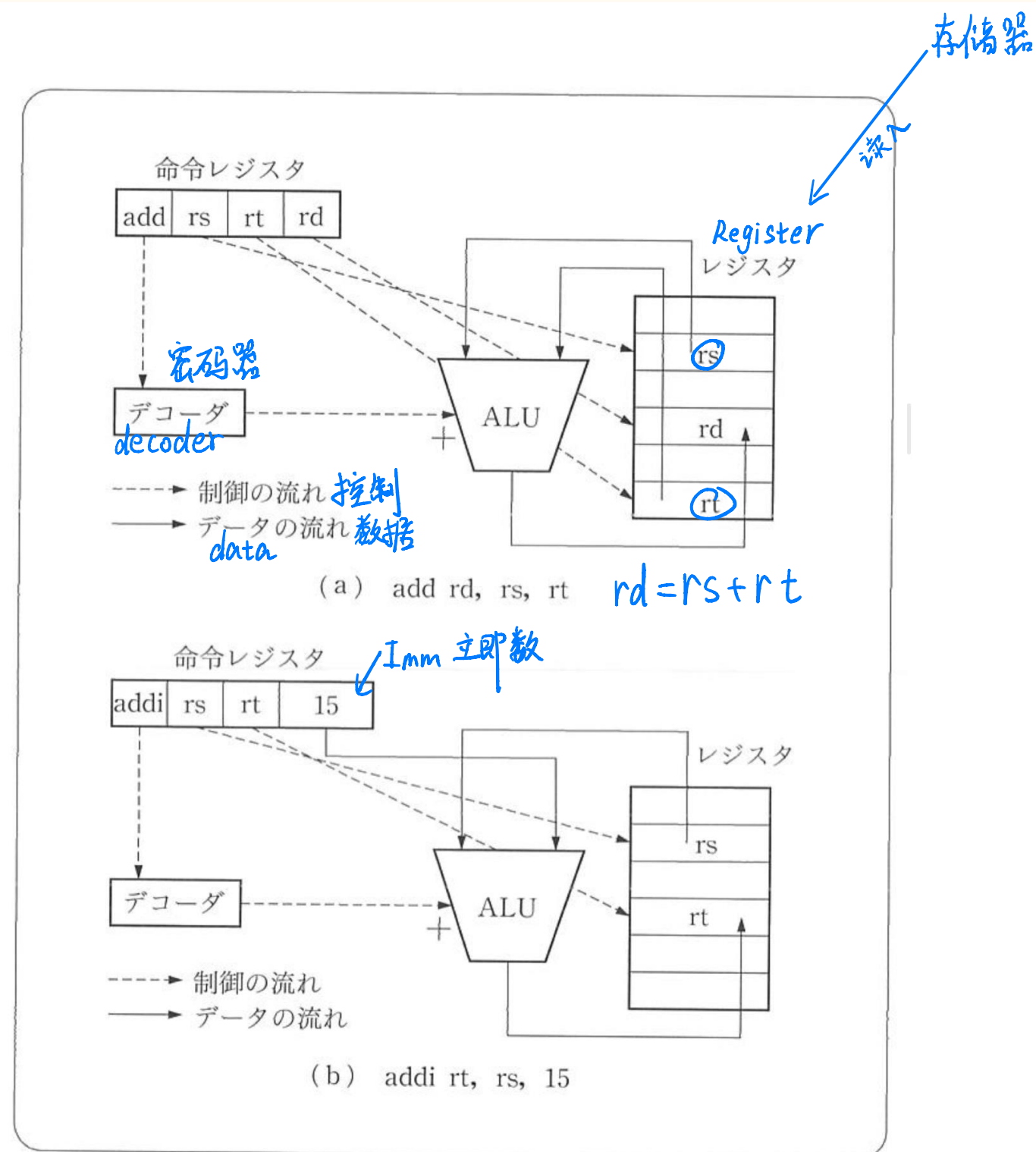
Instruction 1: 00225022 op 0000 0000 0010 0010 0110 0000 0010 0010
 Instruction 2: 01425825
 Instruction 3: 2141fff0
 Instruction 4: ad420080

rs=1 rt=2 rd=10 shamt=0 34 func
 ↓
 sub 1,2,10

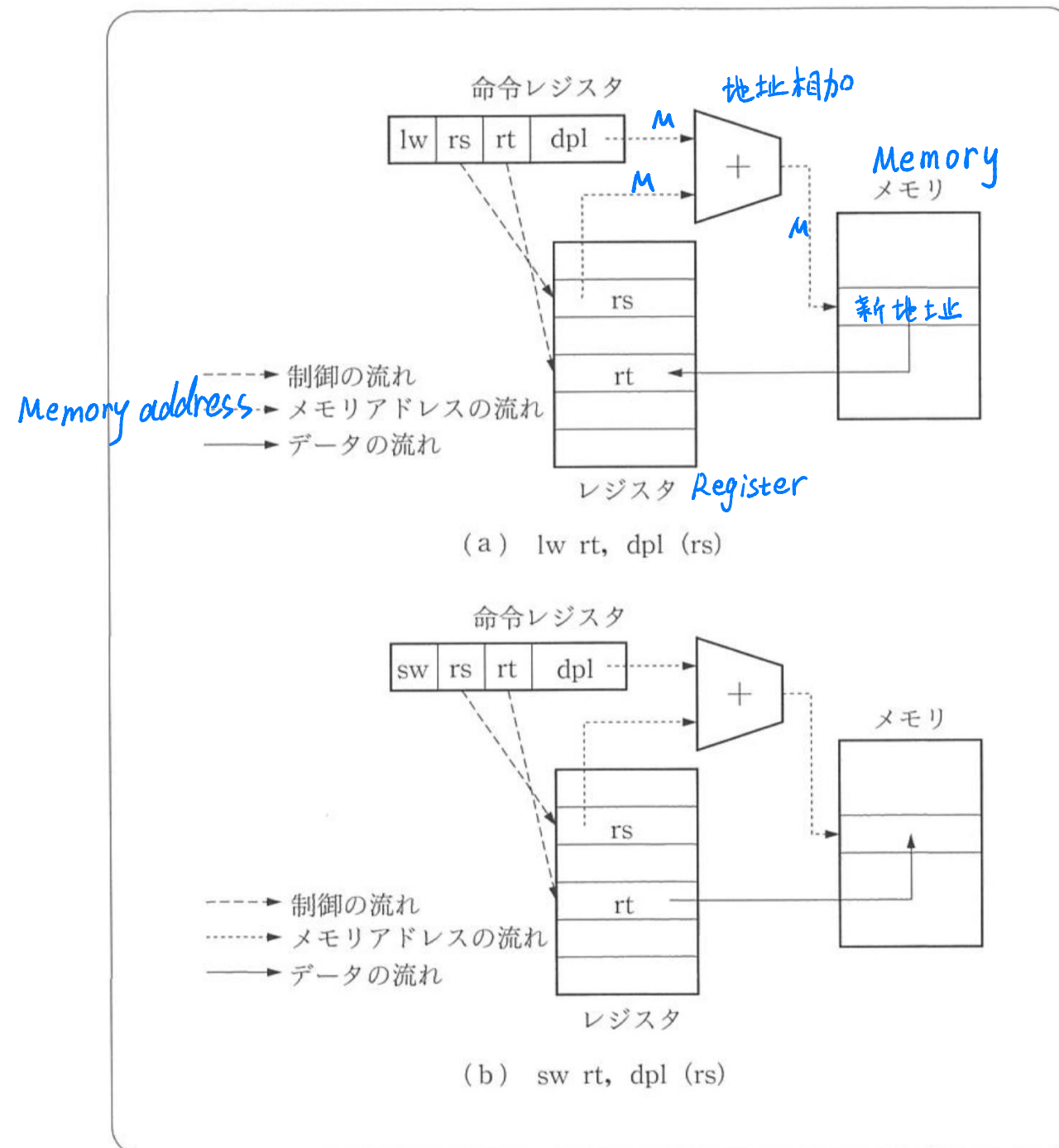
図 (a) 機械語命令列 (16 進数)

Figure (a) Sequence of machine code instructions (hexadecimal).

MIPS格式下的汇编语言



MIPS格式下的汇编语言



MIPS格式下的汇编语言

跳转指令

① $\text{beq } r_1 \ r_2 \ I_k$ 第k条指令
if $r_1 = r_2$ then jump to I_k

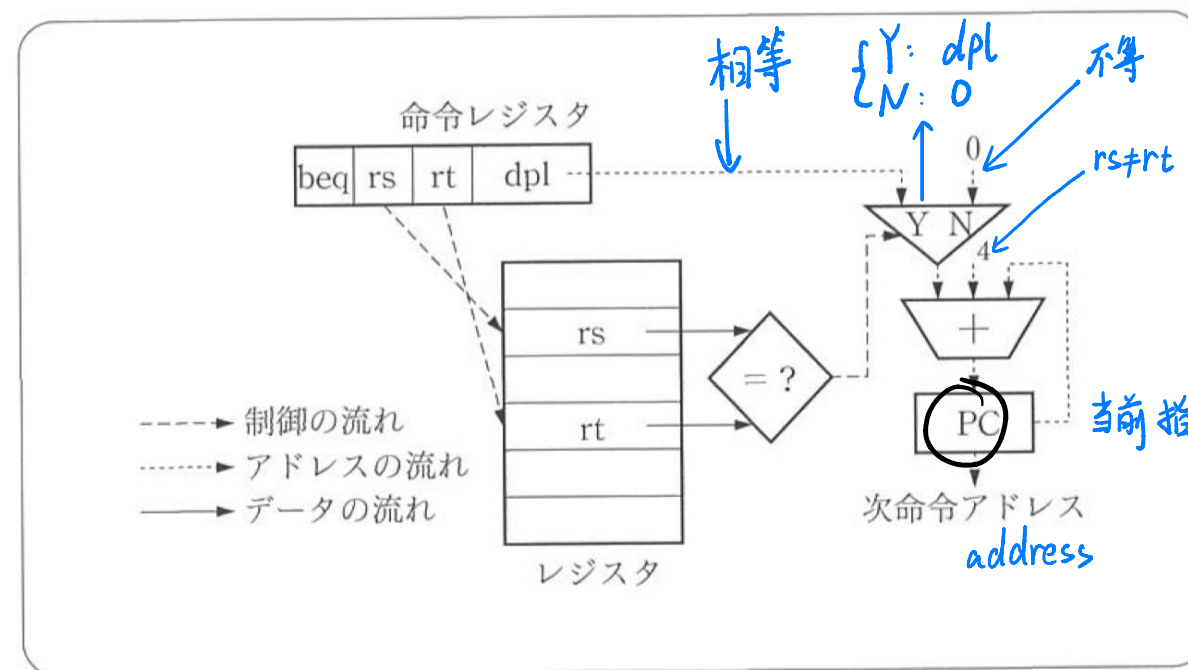
branch of equal

PC(指针)表达

全相邻指令差4

$I_1: \text{beq } r_1 \ r_2 \ 4$ 额外偏移量
+4
 I_2
+0
 I_3

$r_1 = r_2$, $+8 \rightarrow I_3$
 $r_1 \neq r_2$, $+4 \rightarrow I_2$



② bnz branch not zero

③ bz branch if zero

数据表示和运算

计算机组成原理 第二讲



尚岸塾

本讲主要内容

1. 计算机性能指标
2. 定点数的编码表示
3. 定点数的加减、乘法运算
4. 小数的二进制表示与移位运算
5. 浮点数的表示
6. 浮点数的加减运算

计算机性能指标

计算机性能指标

- 发送请求 → 返回结果
1. 响应时间：完成任务的总时间 $\text{CPU时间} + \text{等待时间}$
2. 吞吐率：单位时间内完成的任务数 与存储器紧密相关 (多任务, 不断读写)
3. 时钟周期/时钟频率 GHz 周期 = $\frac{1}{\text{频率}}$
4. CPI: clock cycle per instruction
- 平均执行一条指令的耗时 = $\text{CPI} \times \text{时钟周期}$
- 运算器 I/O, 中断...

定点数的编码表示

定点数的编码表示

定点^{小数点}数：范围小，精度差

用固定整数位数表示实数的格式 隐藏小数点，在固定位置

★浮点数：范围大，精度高，速度慢

用浮点方式表示实数的格式

定点数的编码表示

进位计数法：

在进位计数法中，每个数位所用到的不同数码的个数称为基数。

一个 r 进制数 $(K_n K_{n-1} \dots K_0 K_{-1} \dots K_{-m})$ 的数值可表示为：

$$K_n r^n + K_{n-1} r^{n-1} + \dots + K_0 r^0 + K_{-1} r^{-1} + \dots + K_{-m} r^{-m} = \sum_{i=n}^{-m} K_i r^i$$

式中， r 是基数， r^i 是第 i 位的权重（整数位最低位规定为第0位）； K_i 的取值可以是 $0, 1, \dots, r-1$ 共 r 个数码中的任意一个。

二进制

十进制

十六进制

定点数的编码表示

在计算机系统内部，所有的信息都是用二进制进行编码的：

1. 可以使用两个稳定状态的物理器件表示；
2. 0, 1与逻辑值“真”、“假”对应，方便实现逻辑运算；
3. 可以方便地使用逻辑门电路实现算术运算。

定点数的编码表示

无符号数：

定点数的编码表示

有符号数：

原码

反码

补码

定点数的加减、乘法运算

See 数电.

定点数的加减、乘法运算

定点数的加法运算：

有符号数：注意溢出 overflow

定点数的加减、乘法运算

定点数的减法运算：

小数的二进制表示与移位运算

小数的二进制表示与移位运算

小数的二进制表示:

见数电

小数的二进制表示与移位运算

定点数的移位运算：

ALU { 算术
逻辑
移位
⋮

$(122.2)_{10}$ 右移 n 位，将数相对于“.”右移 n 位 $\rightarrow (1.222)_{10}$ ，缩小 $(\div 10^n)$

逻辑移位：无符号数，添0 $01010011 \xrightarrow{\text{左移1}} 10100110$

算术移位：有符号数，符号位不移动

正：原反补相同，补0
负：{ 原：补0
反：补1
补：左移补0 右移补1

原： $(-53)_{10}$ 10110101
左移1位 $\rightarrow 11101010 = (106)_{10}$ 丢0
右1 $\rightarrow 10011010 = (-26)_{10}$ (丢1)
左1 $\rightarrow (-84)_{10}$ (丢1) 差错

丢0：不影响
丢1：{ 右：精度
左：差错

浮点数的表示

浮点数的表示

广义上的表示:

浮点数的表示

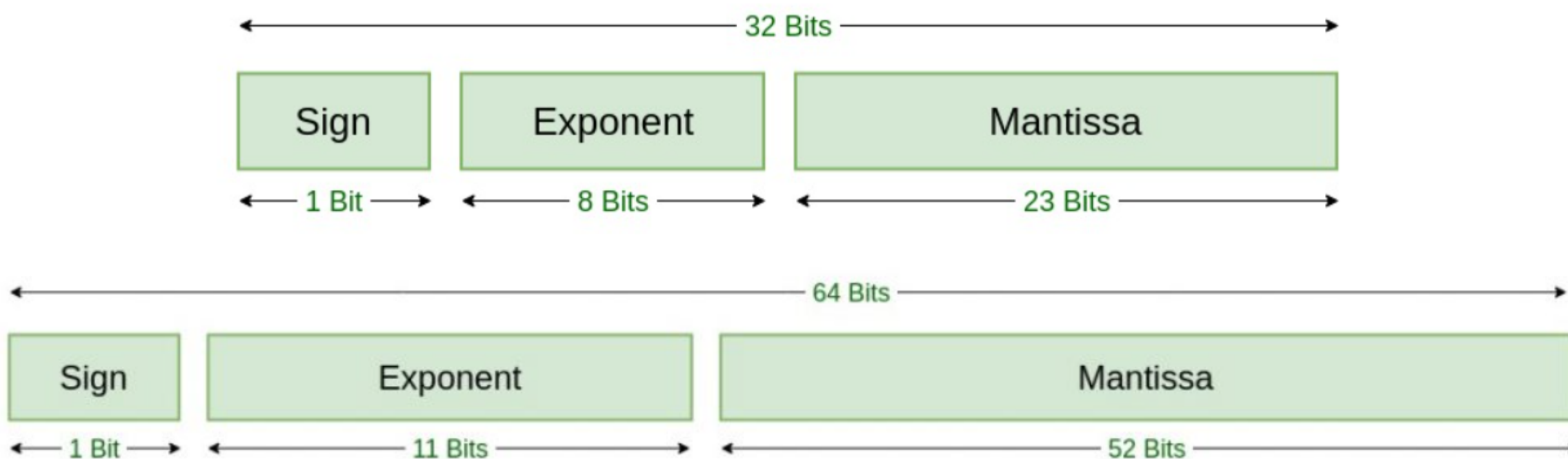
规格化 小数点前只有一位非0数

科学计数法: 6.02×10^{21} ← 阶码 E
 ↑ ↑
 尾数 M 基数 r

对于二进制, $M = \frac{1}{2} . \text{---}$
 ↑
 只能为1

浮点数的表示

IEEE 754 :



浮点数的表示

IEEE 754 :

1. 二进制
2. 移位使最高位为1
3. 隐藏最高位1
4. 计算阶码
5. 写出结果

浮点数的表示

IEEE 754 :

(2) IEEE 754 規格の単精度2進浮動小数点数について, 以下の間に答えよ.

Answer the following questions on IEEE 754 single-precision binary floating point numbers.

(a) -9.375 を IEEE 754 規格の単精度 2 進浮動小数点表現で表せ.

Express -9.375 in IEEE754 single-precision binary floating point representation.

(b) 次の IEEE754 規格の単精度2進浮動小数点数の加算の結果を IEEE754 規格の単精度 2 進浮動小数点表現で示せ. 丸めモードは「最近接丸め」を仮定せよ

Show the result of the following addition of IEEE754 single-precision binary floating point numbers in IEEE754 single-precision binary floating point representation. Assume that the rounding mode is "round to the nearest".

$$0_10100000_11010000000000000000 + 0_10001100_01011000000000000000$$

浮点数的加减运算

浮点数的加减运算

1. 计算阶码
2. 移位
3. 0舍1入
4. 尾数相加减

总结

作业

作业

(1) 2進表現について、以下の問に答えよ。

(a) 次の数を8ビットの2の補数表現で表せ。

(i) +36

(ii) -24

(b) 次の8ビットの2の補数表現の2進数を8ビット符号付き絶対値表現に変換せよ。

(i) 10101111

(ii) 11110000

(c) 次の8ビットの2の補数表現の2進数体系での加算および減算の結果を示せ。

(i) 10101111+11110000

(ii) 10101111+10101111

(iii) 10101111-11110000

(iv) 11110000-10101111

(d) 8ビットの2の補数表現の2進数体系で10101111を2ビット算術右シフトした結果を示せ。

$$\begin{array}{r} 10101111 \\ + 60010000 \\ \hline 10111111 \end{array}$$

(1) Answer the following questions on the binary number system.

(a) Express the following numbers in the 8-bit two's complement representation.

(i) +36 00100100 ✓

(ii) -24 11101000 ✓

(b) Convert the following 8-bit two's complement binary numbers into the 8-bit sign-and-magnitude representation.

(i) 10101111 11010001 ✓

(ii) 11110000 10010000 ✓

(c) Show the results of the following additions and subtractions in the 8-bit two's complement binary number system.

(i) 10101111+11110000 10011111

(ii) 10101111+10101111 overflow ✓

(iii) 10101111-11110000 01111111

(iv) 11110000-10101111 01000001

(d) Show the result of the 2-bit arithmetic right shift operation on 10101111 in the 8-bit two's complement binary number system.

>> 2

$$\begin{array}{r} 10101111 \\ \hline 11101011 \end{array}$$

算術右シフト

$$\begin{array}{r} 10101111 \\ 11110000 \\ \hline 11001111 \end{array}$$

(+16)

$$\begin{array}{r} (00010000)_2 \\ \hline 11101111 \end{array}$$

-16

$$\begin{array}{r} (10010000)_2 \\ \hline 11100000 \end{array}$$

$$\begin{array}{r} 10101111 \\ \downarrow \\ (11010001)_2 \\ 2^6 + 2^0 + 1 \\ = -(32 + 16 + 1) \\ = -49 \end{array}$$

$$\begin{array}{r} 11110000 \\ \downarrow \\ (00011111)_2 \\ (00100001)_2 \\ \downarrow \\ -2^0 = -16 \end{array}$$

$$-49 + 16 = -33$$

作业

(2) 次の6ビットの2の補数表現の2進数に対する乗算において全ての部分積を明示することにより2次のブースのアルゴリズムを説明せよ。

被乗数: 011101, 乗数: 010110

(2) Explain radix-4 modified Booth's algorithm by showing all the partial products of the multiplication for the following pair of 6-bit two's complement binary numbers.

Multiplicand: 011101, Multiplier: 010110

011101 × 010110 正 × 正

$$\begin{array}{r}
 011101 \\
 \times 010110 \\
 \hline
 000000 \\
 011101 \\
 011101 \\
 000000 \\
 011101 \\
 \hline
 0010011110
 \end{array}$$