第二章 指令系统

.

※本章主要内容

(1) 指令系统概述　　指令系统的概念、设计概述

(2) ISA结构设计　　结构分类，结构及相关参数设计

(3) 数据表示设计　　设计方法、设计举例

(4) 指令功能设计　　CISC功能设计及优化，RISC功能设计及优化

(5) 寻址方式设计　　编址方式设计、寻址方式设计

(6) 指令格式设计　　操作码设计及优化，指令格式设计及优化，指令系统分析

※总体要求

掌握指令系统的组成，理解各部分的设计方法

（要设计什么）　　（怎么设计）

第1节　指令系统概述

※主要内容：ISA所含内容，ISA设计的目标、准则、过程

一、指令系统的概念

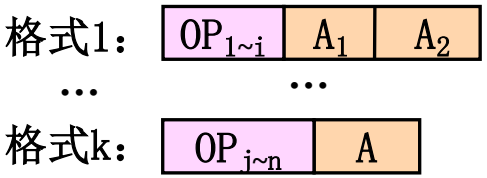
1、指令系统的定义

*机器指令：　指硬件可直接识别和执行(实现功能)的命令

表示— 一定格式(指令格式)的二进制编码

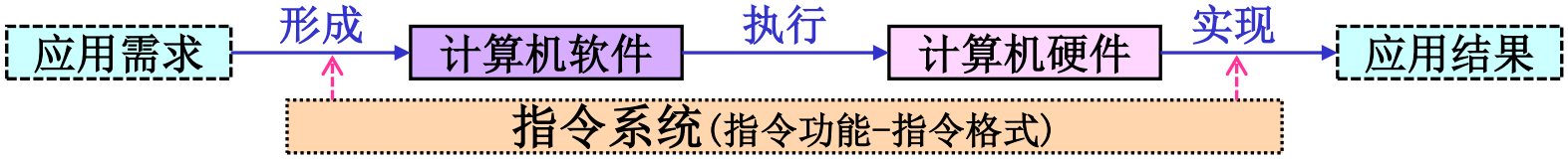
*指令系统：　指所有机器指令的集合，即指令集(Instruction Set)

└→反映硬件所有功能



多条指令共用一种格式
不同指令通过OP码标识

实质— 指令功能-指令格式(硬件-软件)的约定



思考：指令系统与ISA的关系？

是ISA的主要部分，常混用！
(ISA还约定了硬件子系统、软件管理的软件接口)

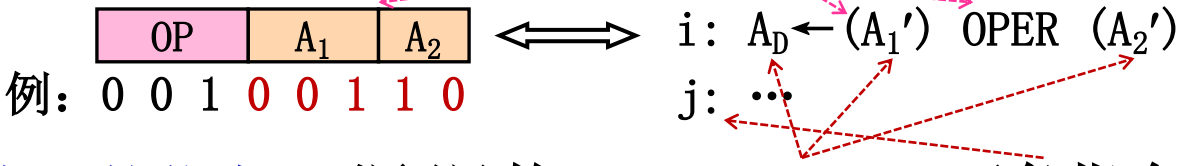
2、指令格式的组成

***指令功能所含信息：操作类型、源/目的OPD、下条指令地址**

***指令格式的组成：** （目标是表示指令功能）

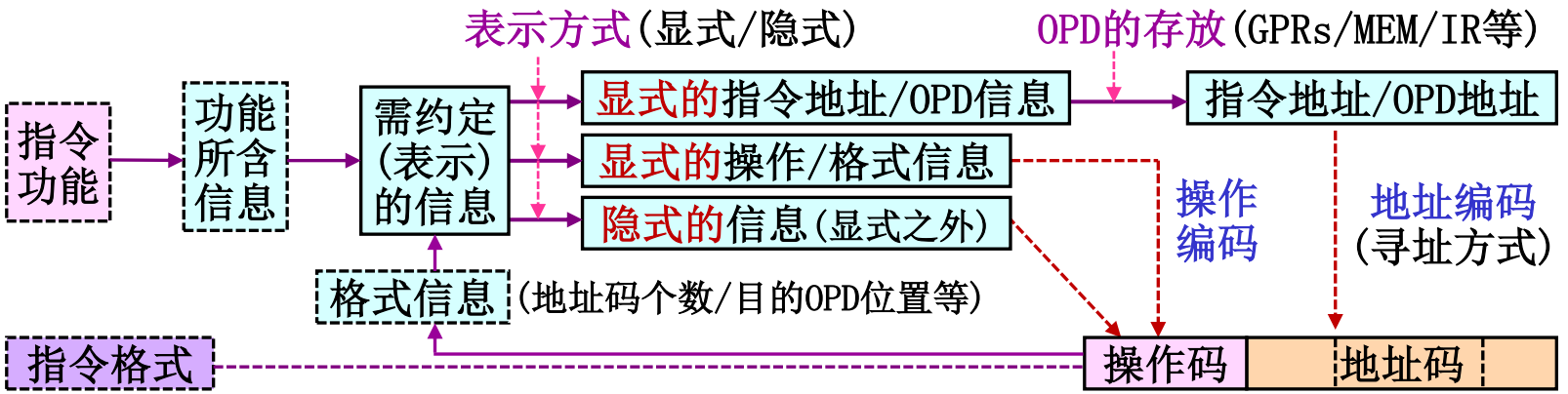
操作码表示的信息— 操作类型、格式信息(地址码个数+目的OPD位置等)

标识不同指令的功能



地址码表示的信息— 源/目的OPD(地址形式)、下条指令地址

与指令功能的关联— 编码表示、尽量短(→显式/隐式表示+缩短编码)



思考：信息隐式表示的条件？必须显式表示的信息？隐式信息如何表示？

3、指令系统所含内容

***所含内容：**

ISA结构—类型(基于OPD存放部件)，指令字结构(长度/地址码个数)

数据表示—支持的数据类型(格式/参数)

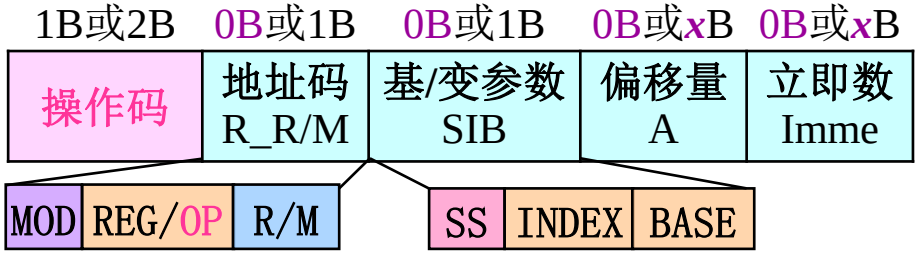
每条指令为数据表示的子集

指令功能—支持的操作类型(操作功能+数据类型)

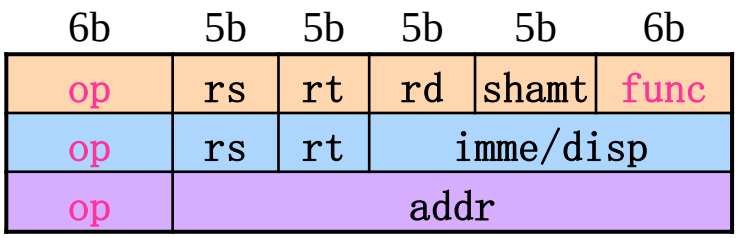
寻址方式—支持的寻址单元及范围、地址形成规则

(编址方式&地址空间)　(寻址方式)

指令格式—指令信息的表示(格式/编码)



IA32的指令格式



MIPS32的指令格式

4、性能指标

*性能指标：指令系统对软件&硬件的支持程度，
包括程序的代码效率、执行效率

*代码效率：包括指令条数及指令字长，应尽量小

减少指令条数—指令功能强、地址码个数多，
寻址方式多、寻址范围大、对称性好等

R←R+M
及M←R+M

缩短指令字长—地址码个数少、长度短，冗余空间小等

例1：32位运算频率较高时，若IS仅支持8/16位运算，则1个32位运算→2个16位运算指令，代码效率不高(指令条数多)

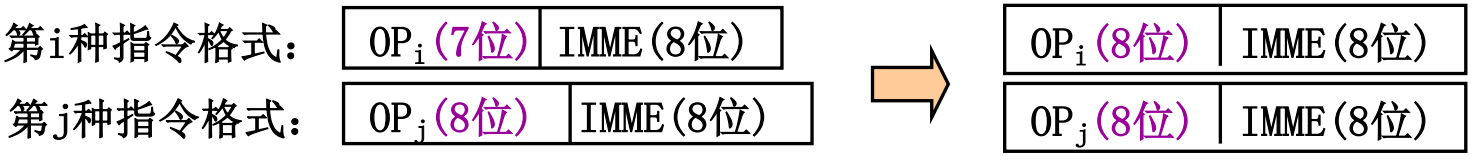
例2：数组操作频率较高时，若IS仅支持基址寻址、不支持变址寻址，则代码效率不高(指令条数多)；不支持基址+变址寻址(如(RB)+(RI))，则代码效率不高(地址码较长)

*执行效率：包括译码时间及执行时间，应尽量少

减少译码时间—指令格式规整、寻址方式并行识别等

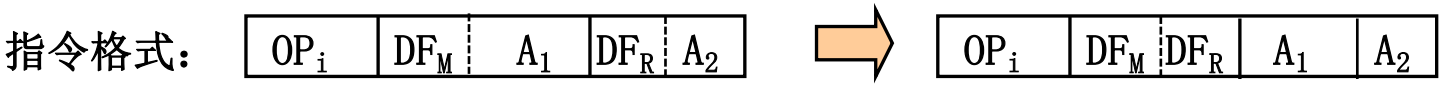
减少执行时间—指令功能弱、OPD访问速度快、地址形成简单

例3—指令格式不规整时，译码速度较慢(需多级译码)



例4—寻址方式位前移/部分缺省，可缩短译码时间；

指令字中MEM_OPD较多时，执行时间长(需多次访问MEM)



$T_{译码}=T_{OP}+T_{DFM}+T_{DFR}$　　$T_{译码}=T_{OP}+\max(T_{DFM},T_{DFR})$

$T_{执行}=T_M+T_R+T_{OP}+T_R$或$T_M$，$A_1$为目的OPD时$T_{执行}$较长

*代码效率与执行效率的冲突处理：

采用折中方法，用CPU时间($=I_N*CPI*T_C$)评价

二、指令系统的设计概述

1、设计的基本原则　（关注对性能指标的影响）

完整性— 功能齐全、使用方便

（如不支持NOT指令[频率较低]时，需支持NOR、恒0寄存器）

规整性— 相似的OP、OPD等有相同的约定　（如OP编码、REG使用）

正交性— 指令字中不同字段相互独立　（如OP/Addr字段）

兼容性— 预留部分操作码，便于IS扩充　（如系列机）

2、设计的基本过程

ISA结构设计— 确定结构类型，指令字结构、显式OPD个数

数据表示设计— 确定支持的数据类型

指令功能设计— 确定支持的OP类型

寻址方式设计— 确定部件的编址方式及地址空间、寻址方式

指令格式设计— 确定各条指令的格式及编码

设计优化— 包含各个环节，多次迭代

第2节　ISA结构设计

本为IS结构，用ISA结构代替
(避免与ISA混淆)

※主要内容：结构分类、 结构及相关参数设计

一、ISA结构分类

*分类方法：按操作数(OPD)的存放部件分类　←指运算类指令OPD

*结构类型：（趋势—R-R风格的通用寄存器型）

堆栈型—OPD放在栈中(栈在MEM中)，代码效率低/OPD访问慢

累加器型—OPD放在AC及MEM中，指令字长长/OPD访问较慢

通用寄存器型—OPD放在GPR中，指令字长短/执行速度快/…，

风格有R-M型(CISC)、R-R型(RISC)

堆栈型	累加器型	R-M风格的GPR型	R-R风格的GPR型
push X	load X ; AC←M[X]	load R1,X ; R1←M[X]	load R1,X ; R1←M[X]
push Y	add Y ; AC←(AC)+M[Y]	add R1,Y ; R1←(R1)+M[Y]	load R2,Y ; R2←M[Y]
add	store Z ; M[Z]←(AC)	store R1,Z ; M[Z]←(R1)	add R3,R1,R2 ; R3←(R1)+(R2)
pop Z	注：AC是唯一的数据REG	注：运算指令中可不含M_OPD	store R3,Z ; M[Z]←(R3)

*相关参数：指令字结构、显式OPD个数

二、ISA结构设计

设计基础—应用需求(多个高级语言程序)，性能、性/价需求

设计内容—ISA结构，指令字结构、显式OPD个数

*ISA结构的确定：

类型—堆栈型、累加器型、通用寄存器型(R-M或R-R)

设计— 趋势为R-R风格的GPR型，如MIPS、RISC-V

| └←利于并行处理(指令功能简单)

└←硬件成本下降

*指令字结构的确定：

类型—定长指令字、变长指令字 ←指令字长=*k*存储单元长度

设计— R-R型结构采用定长结构，R-M型结构采用变长结构

└←提高对硬件的支持程度 └←提高对软件的支持程度

思考：为何R-M型结构不宜采用定长指令字结构？

R-M型指令集支持R-R型、R-M型指令，R-OPD地址较短、编码可一种，M_OPD数据结构类型较多、一种编码时效率较低

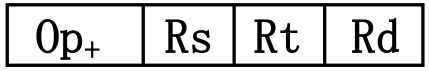
.

*显式OPD个数的确定：　（主要指运算型指令[其余指令OPD较少]）

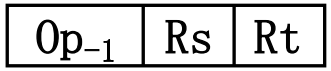
类型—堆栈型0个，累加器型1个，GPR型≥2个

例1—下列指令字中，显式OPD个数分别是多少？

Rd←(Rs)+(Rt)



Rt←(Rs)-1



设计— 基于ISA结构(通常RISC＞2、CISC＝2)　←提高指令效率

结构类型	显式OPD个数	MEM型OPD数	示例
R-R	3	0	MIPS、PowerPC、ARM
R-M	2	1	80x86、Motorola 68000
M-M	2、3	2、3	PDP-11、VAX

例2—R-R型ISA支持的显式OPD≤2个时，Rd←(Rs)+(Rt)需用几条指令实现？为什么常为3个OPD？

Rt←(Rs)+(Rt)及Rd←(Rt)；RISC常采用定长指令字格式（提高执行效率），load/store指令含MEM地址、指令字较长，其余指令不采用3个OPD时冗余空间较多。

第3节　数据表示设计

※主要内容：数据表示的设计方法、设计举例

一、数据表示的设计

1、数据表示与数据结构

*定义：数据结构—指软件能够直接识别和引用的数据类型

<值集, 操作集>→┤

数据表示—指硬件能够直接识别和引用的数据类型

*两者关系：数据表示是数据结构的子集

数据结构 —直接使用法(x%)→ 数据表示

—软件映像法(1-x%)→

软/硬件间语义差别

*数据表示所含内容：表示方法(格式/进制/编码)＋数据长度

思考：数据表示为什么要包含长度？

不同数据类型的值域不同，硬件处理时序固定长度

2、数据表示设计

设计基础—需求程序，性能及性/价需求

设计内容—拟支持的数据类型及其参数(种类、格式/参数)

设计目标—性/价较高，性能关注存储效率、计算速度

思考①：ASCII存放在字节、双字节中时的存储效率分别是多少？
若单精度FPU每步时延为1t，则双精度FPU的时延至少是多少？

***数据表示的常见类型：**

基本数据表示—定点数、浮点数、逻辑数、字符等

高级数据表示—向量、树、链表、指针等

思考②：向量数据如何表示？向量数据表示有哪些属性？

程序：
```
int A[256],B[256],C[256],i;
for (i=16;i<48;i++)
   C[i]=A[i]*B[i];
```

程序：
```
Vint A[256], B[256], C[256];
//一维标量数组(含分量类型及个数)
ADDV C[16],A[16],B[16],32
```

思考③：支持向量表示后，哪些方面的性能有提升？需增加哪些硬件？

思考①：分别为7/8、7/16；至少为5t（位数运算增加1t）
思考③：指令条数减少、向量处理速度提高（无需多次取指/译码），
需增加向量处理部件、向量寄存器/存取缓冲器

*数据表示的设计：

方法— 需求程序分解为若干基本操作及基本数据类型，

统计各种基本数据类型的使用频率，

选择拟支持的数据类型及其参数

} 称为频带分析法

结果— 支持频率较高的数据类型及其参数(常有多种)

└←重点关注经常性事件原理　　└←存储效率好

影响因素— 性能、性/价需求

例—若标量部件占硬件成本的5%，向量部件的速度、成本分别为标量部件的10倍、3倍，测试程序中向量操作占10%，则支持向量表示后的性/价是否有提高？

解：设测试程序执行时间＝t，硬件成本＝s，则性/价＝(1/t)/s；

新的执行时间＝ 0.9t+0.1t/10＝0.91t，

成本＝ s+0.05*3s＝1.15t，　←标量部件还需要存在

性/价＝ (1/0.91t)/1.15s＝0.96/(t*s)

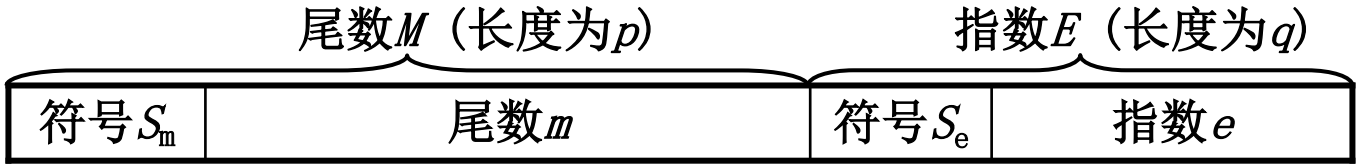
二、数据表示的设计举例—浮点数

1、浮点数的表示

*表示方法：纯小数尾数＋纯整数指数(阶)　←隐含小数点

$N=M\times r_m^{E}$，$M=(-1)^{S_m}\times 0.m_{-1}\cdots m_{-p'}$，尾数的基为$r_m$；

$E=(-1)^{S_e}\times e_{q'}\cdots e_0$，指数的基为$r_e$



思考：p′与p、q′及q的关系？　$p-1=p'\cdot\log_2 r_m$，$q-1=q'\cdot\log_2 r_e$

*格式参数：尾数和指数的码制、数制(r_m及r_e)、长度(p及q)

*设计内容：格式参数设计、操作处理设计

设计基础—需求程序，表示范围及精度需求(如≥N及≥δ)

2、浮点数的格式参数设计

***码制的确定：** （*M*及*E* 的编码）

设计目标—便于浮点运算的硬件实现

选择范围—原码、补码、移码等

设计结果—尾数常为原码/补码，指数常为移码(便于对阶)

***数制的确定：** （r_m及r_e的取值）

规格化数个数/浮点数个数
$=(r_m-1)r_m^{p'-1}/r_m^{p'}=(r_m-1)/r_m$

设计目标—表(示)数的范围、精度、效率均较好

$=\pm(1-r_m^{-1})r_m^{Emax}$ →┘　　└←尾数最低位的1/2, 即$=r_m^{-(p'-1)}/2$

选择范围—2、4、8、… （2^n为无冗余编码）

r_e设计—分析：整数的表数范围&精度<u>不受基影响</u>(如$2^8=4^4=16^2$)

结果：$r_e=2$(最易对阶)

r_m设计—分析：r_m↑时范围↑、精度↓、效率↑；q↑时范围↑

结果：通常$r_m=2$(保精度)、多种q(扩范围)及p(保效率)

*数码长度的设计：

设计目标—满足实数的表示需求(范围≥N、精度≥δ)

选择要求—$p+q=k\times$存储单元长度

设计方法— 根据N及δ，计算对应的p、q，并适当调整

设计结果— 多种p及q，采用IEEE 754标准(性能优、兼容性好)

思考①：同一类数据有多种表示长度的好处？ 可提高存储效率、计算速度

思考②：IEEE 754的格式参数(码制/进制/长度)？

3、浮点数的操作处理设计 (约定最低要求)

设计目标—减少数据在运算过程中的精度损失

设计内容—附加位(=保护位+舍入位)位数、舍入方法

～左规精度→┘ └←～运算精度

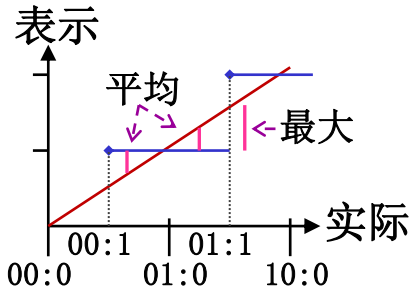
*附加位位数的选择：满足结果精度要求(大概率情况)

IEEE 754为≥1位保护位+1位舍入位

└←±/×需1位、÷需0位

*舍入方法的选择：

←舍/入是基于数轴的



选择依据—最大误差、平均误差，实现成本

$\max\{|值i_{表示}-值i_{实际}|\}$　$\Sigma(值i_{表示}-值i_{实际})$

选择范围—有截断、恒置1、舍入法

IEEE 754有向上舍入(向+∞舍入)、向下舍入(向-∞舍入)、

向零舍入(舍)、就近舍入(4舍/6入/5成偶数)

└←又称向偶数舍入法

	-x:ab				+y:ab			
	ab=11	ab=10	ab=01	ab=00	ab=00	ab=01	ab=10	ab=11
向上舍入	-x			-x	+y	+(y+1)		
向下舍入	-(x+1)					+y		
向零舍入	-x					+y		
就近舍入	-(x+1)	x为偶数：-x x为奇数：-(x+1)	-x			+y	y为偶数：+y y为奇数：+(y+1)	+(y+1)

选择结果— 舍入法最好

IEEE 754默认采用就近舍入法

第4节　指令功能设计

※主要内容：CISC的功能设计及优化、RISC的功能设计及优化

设计基础—需求程序，已确定的ISA结构、数据表示

设计内容—拟支持的操作类型(OP功能+OPD类型)

思考：操作类型为何要区分OPD类型？

*常见的指令功能：

操作—

分类	操作类型	数据类型相关性
算逻运算	整数加/减/乘/除等，逻辑数与/或/非等	√
浮点运算	浮点加/乘等	√
数据传送	Load/Store，In/Out	×
转移控制	分支/跳转/调用/返回/自陷等	√/×
系统控制	OS调用、虚存管理等	×
十进制	十进制加、乘等	√
字符串	字符比较，字符串拷贝、移动、比较等	√
图形	象素操作、压缩/解压操作等	√

操作数— 指运算时的数据类型，OPD集合≤数据表示集合

└←～硬件成本

如：数据表示支持支持8/16/32b整数，ALU仅支持32b运算

思考：不同**OPD**类型的相同操作的实现方法不同，需用不同的**OP**码表示，如整数加、浮点加

一、CISC的指令功能设计及优化

1、CISC的指令功能设计

*CISC目标：侧重强化指令功能，减少目标代码长度

*设计方法：

①需求程序分解为若干基本操作及数据表示

数据表示设计已完成

②统计各基本操作类型的使用频率

③选择拟支持的基本操作

频带分析法

*设计结果：支持频率较高的基本操作

例1—数据表示的设计结果支持BCD数，需求程序中各基本操作及使用频率如下，请进行指令功能设计。 （忽略OPD设计）

操作类型		频率	操作类型		频率	操作类型		频率	操作类型		频率
算术	加减	0.150	传送	取	0.060	浮点	加减	0.060	十进制	加减	0.008
	±1	0.050		存	0.050		乘除	0.039		乘除	0.007
	乘除	0.100	控制	分支	0.100		±1	0.001		调整	0.005
逻辑	与	0.040		跳转	0.050	图形	叠加	0.015	字符串	拷贝	0.004
	或	0.040		调用	0.020		反色	0.003		比较	0.004
	非	0.005		返回	0.020		压缩	0.001		移动	0.002
	异或	0.020		自陷	0.015		解压	0.001	系统		0.140

指令功能设计结果：支持频率≥1%的操作、特殊操作(如调整)

说明--①低频率操作可用软件实现，如逻辑非、浮点±1

②可分解的操作无需统计，如图形反色等

③特殊操作必须支持，如调整(二进制运算←→BCD码)

(保持完整性)　　(15H+06H=1BH→21H)

替代操作?

2、CISC的指令功能优化

静态频度分析及动态频度分析

*优化方法：

①统计各指令操作在代码中及执行时的使用频率，

②根据指令(串)的使用频率，增强指令功能、增设新指令

*面向目标程序改进：

增强指令功能，减少代码条数；　　如：REP，新的寻址及OPD类型

设置新指令，减少代码、提高速度　如：复杂功能，CMP+Brn合并

*面向高级语言和编译程序改进：

设置新指令，缩小语义差别；　如：LOOP、INC

强化对称性，优化代码生成　　如：R/M←R+M，新的寻址方式

*面向OS改进：

设置专用指令，优化OS的执行效率

如：进程同步、信息保护、工作状态切换等专用指令

二、 RISC的指令功能设计及优化

1、CISC ISA的主要问题

为了提高代码效率→指令格式、编译、体系结构的复杂化

因为指令功能不平衡→不利于并行处理(如流水线)的实现

因为指令格式复杂→控制器复杂，不利于VLSI的实现

2、RISC的指令功能设计

*RISC目标：侧重简化指令功能，
以提高执行速度、适应并行处理及VLSI

*设计方法：同CISC的功能设计方法

*设计结果：支持频率很高的基本操作 ←CISC为频率较高

例2：对例1，RISC支持使用频率≥2%的操作、特殊操作

3、RISC的指令功能优化

*优化方法：同CISC的统计方法、优化手段

*优化结果：
- 支持特殊操作，以提高效率　　如：BCD数的调整
- 增加指令功能，以支持高级语言和编译程序

  如：增加OPD个数、子功能，利用冗余位实现
- 设置专用指令(特权指令)，以支持OS
- 指令执行在1个时钟周期完成，提高功能平衡性

  如：指令的OPD均为R-R型，Load/Store指令除外

4、RISC与CISC的比较　　($T_{CPU}=I_N \times CPI \times T_C$)

*特征：CISC—强化指令功能，减小了I_N、增加了CPI

RISC—简化指令功能，减小了CPI、增加了I_N

*结果：RISC占优(利于并行处理、VLSI)　　←器件成本下降

作业1：P51—5、PPT—3

第5节　寻址方式设计

※主要内容：编址方式设计、寻址方式设计

一、编址方式设计

设计基础—已确定的ISA结构(部件类型[多种])，需求程序

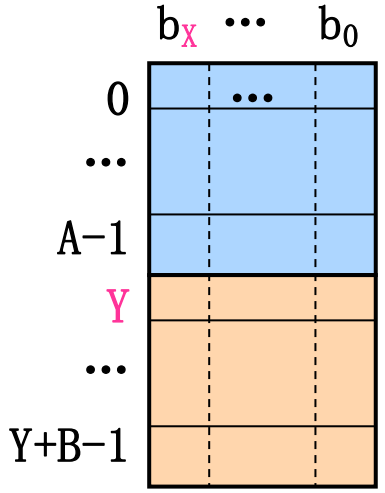
设计内容—

①编址单位—部件单元存放的信息长度(x)

②地址空间—部件单元的个数(A/B)

③编址方式—不同部件地址间的关系(y)

影响部件的识别方法←┘



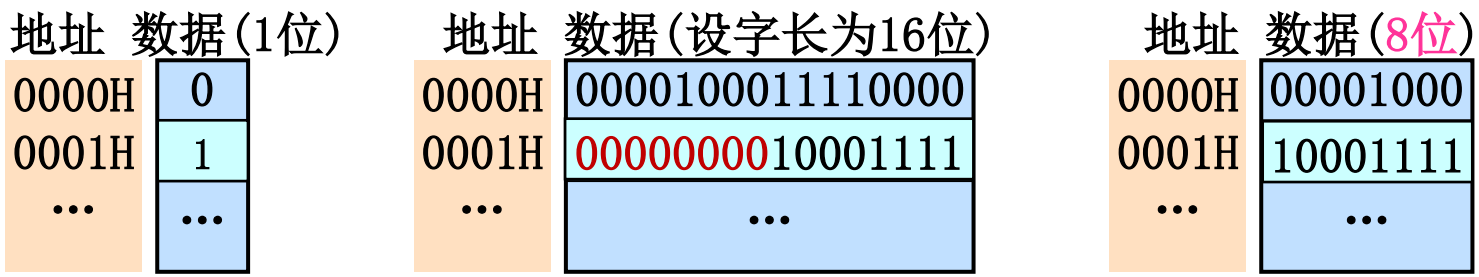
选择依据—

存储效率(数据长度/占单元数)、访问效率(访问次数/OPD)均好

1、MEM的编址单位设计

*数据长度的特征：逻辑数(1位)、其它数据(m种)

*编址单位的类型：二进制位、机器字长、折中长度



*编址单位设计：

类型选择— 常为折中长度 ←重存储效率、轻访问效率

└←可改进[如并行访问]

设计结果— ＝min{所有数据表示的长度}＝2^m位，

通常为字节 ←ASCII码最常用、逻辑数可组合

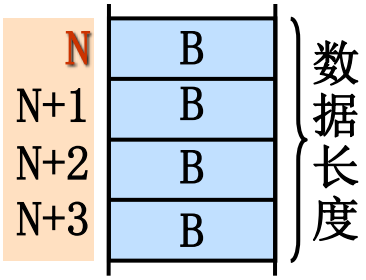
数据存放— 占k个连续存储单元($k \geq 1$)

└→如何放？→端序+对齐问题

*数据存放方式设计：

存放方法—存放在连续的存储单元中

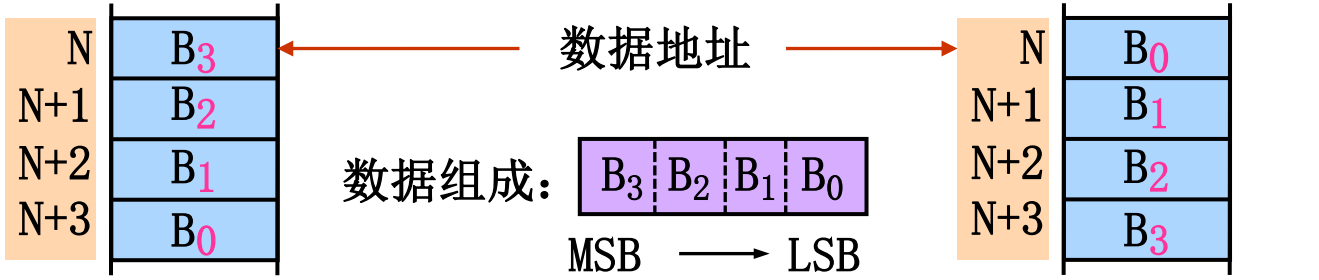
数据地址—用最小单元地址(N)表示



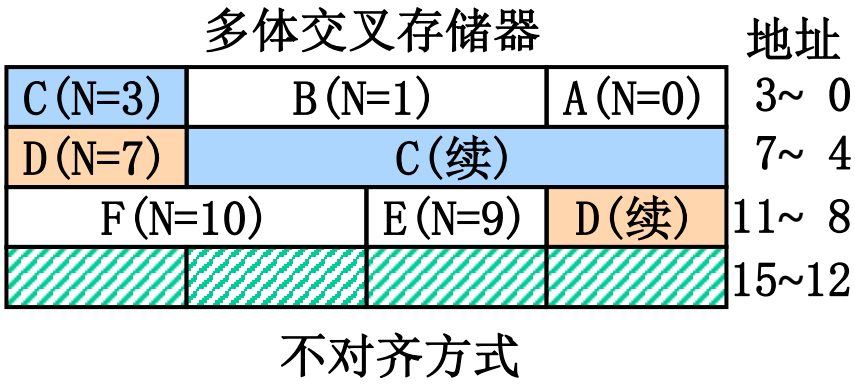
存放次序设计—大端、小端方式，任选一种

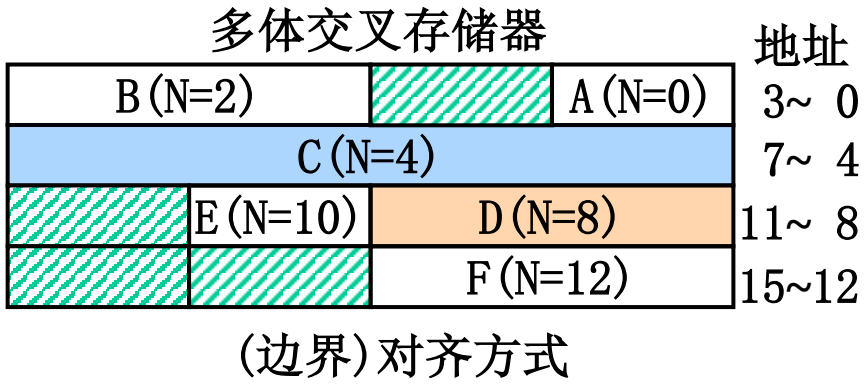
N中内容的约定

N取值的规则



存放位置设计—不对齐、对齐方式，现均为对齐方式





2、其它部件的编址单位设计

***外设的编址单位：与MEM一致** ←面向传输(均通过总线访问)

└→含数据存放方式(端序/对齐)

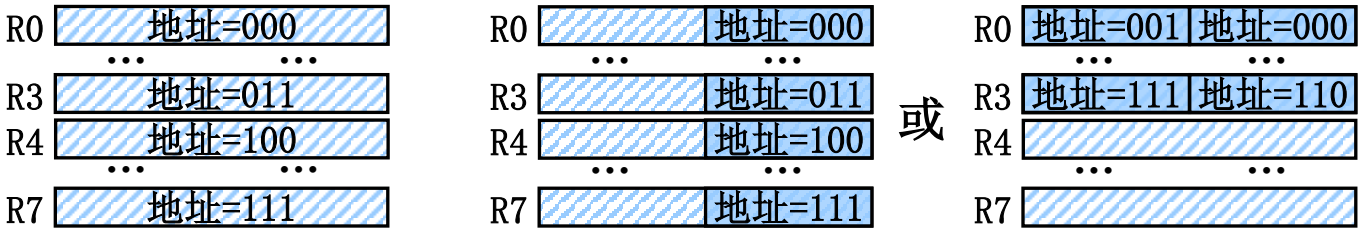
***REG的编址单位：与OPD长度对应** ←面向运算(暂存运算结果)

└←＝ALU运算位数，可有多种

思考①—数据表示有8/16/32位，ALU支持16/32位，REG编址单位？依据？

思考②—REG有多种编址单位时，用多个REGs还是一个REGs实现？

REG组织— 设置一个REGs，通过重命名构成多个地址空间



使用方法— 多个地址空间可同时使用 ←每条指令只使用部分REG

如：$R4_{32位} \leftarrow (R4_{32位}) << (R0_{16位})$

***堆栈的编址单位：常与REG一致** ←操作速度快(无需长度转换)

思考①：16及32位；数据表示关注存储效率，REG编址关注对运算速度的支持(简化OPD长度转换时延)

思考②：一个REGs

3、地址空间设计

***MEM地址空间设计：**

所有需求程序(考虑可扩展性)的最大值(=2^m)　如：ARMv8为48位

***I/O地址空间设计：**

所有需求程序(考虑可扩展性)的最大值(=2^n)　如：IA32为16位

***REG地址空间设计：**　(即确定寄存器的数量)

专用REG数量—取决于OS、组成，无需编址(数量少/操作码指明)

如：控制REG、状态REG等

通用REG数量—取决于编译程序、需求程序，显式编址

(REG分配算法)＋(变量使用特征)

如：IA16为8个、部分通用(有限制)

4、编址方式设计

指不同部件(REG/MEM/IO)地址间的关系(→区分方法)

*编址方式的类型： 独立编址、统一编址 ←堆栈无需编址

思考—不同编址方式对指令系统的要求是什么？

*编址方式的设计：

REG与MEM及I/O之间— 独立编址方式 ←访问路径不同、地址短

MEM与I/O之间— 任选一种 如：x86为独立编址，ARM为统一编址

二、寻址方式

设计基础—需求程序，部件编址方式(单元/空间/相互关系)

设计内容—拟支持的寻址方式及其参数

1、寻址方式种类的确定

*寻址种类：

常见— 立即，REG，直接、REG间接、偏移(基址/变址)、相对等

思考①—i及A均是地址码参数，OPD=(i)、M[(i)]、M[A]、M[(i)+A]、M[(PC)+A]、A时的寻址方式各是什么？存放部件有哪几种？

思考②—基址寻址、变址寻址的区别是什么？适用场景？

思考③—为什么M_OPD有多种寻址方式，而R_OPD只有一种？

其他— 索引[(RB)+(RI)]、比例变址[(RI)*S+A]等

*选择依据：能/否减少指令长度或指令条数，对指令CPI影响小

思考①：寄存器寻址、寄存器间接寻址等，寄存器、存储器、指令寄存器

思考②：寄存器中存放的是基地址/可变地址，适用于存储管理、数组访问（下标需改变）

思考③：MEM地址较长，地址码短是获益较大；REG地址较短，没必要忙乎

*设计方法：形成需求程序中各基本操作的数据&指令寻址方式，

└←种类尽量多

统计各种寻址方式的使用频率，

选择高频率的寻址方式

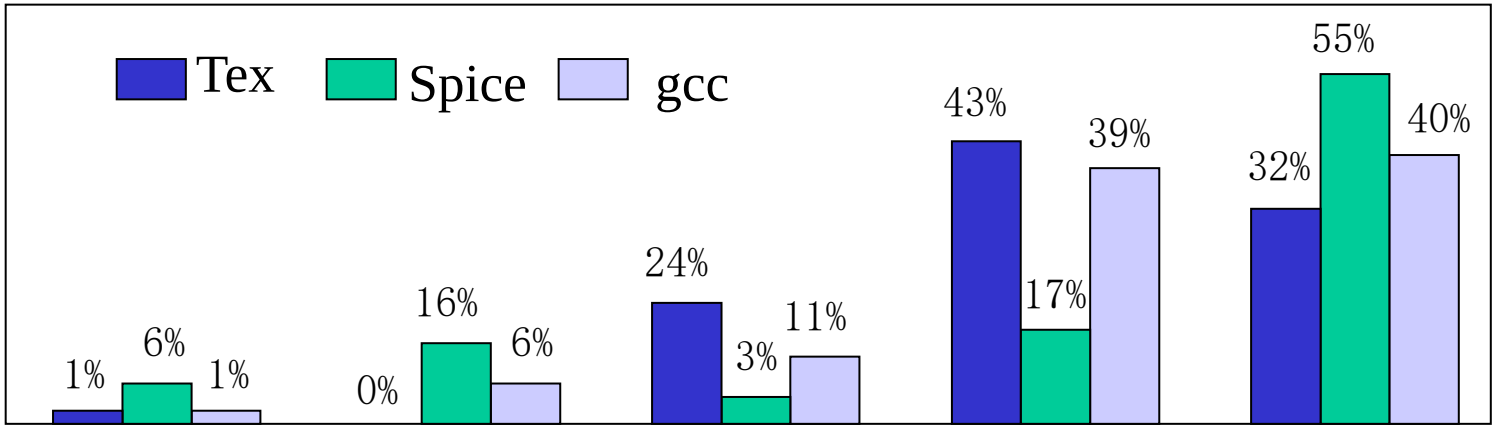
*设计结果：　　（高频率的定义不同）

CISC— 尽量多的寻址方式，增加灵活性　←减少指令长度/条数

RISC— 尽量少的寻址方式，简化硬件　←对CPI影响小

.

例1：VAX计算机中，运行gcc、Spice和Tex基准程序，各种寻址方式的分布如下，该支持哪些寻址方式？



注：偏移寻址—带偏移量的寻址方式，如基址、变址、相对寻址

选择结果：

①偏移、立即、REG间接寻址方式，须支持(高频率)

②其他寻址方式，依据CISC/RISC风格，再选择

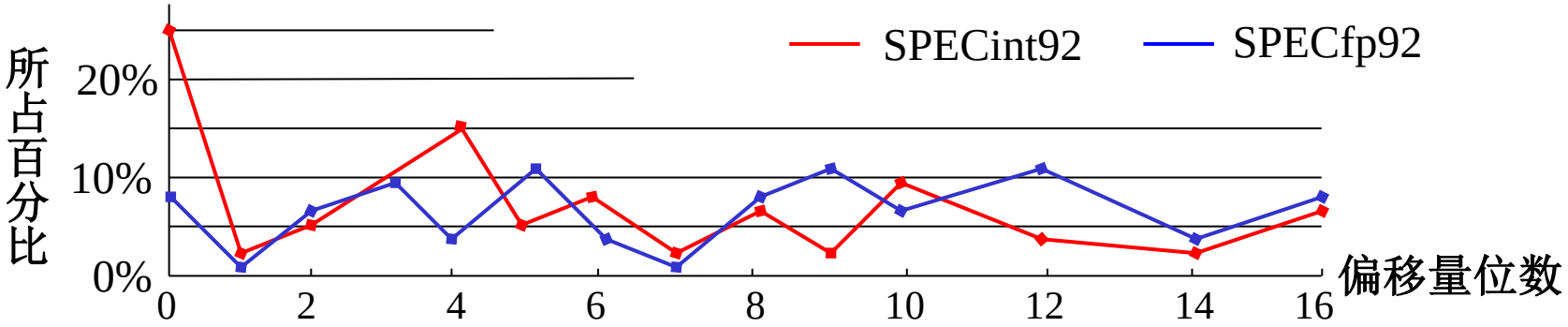
2、寻址方式参数的确定

***设计方法：** （对每种寻址方式分别设计）

统计每个参数不同取值的使用频率， ←即频带分析法

选择满足需求(≥m%)的参数值(可有几种)

例2：某R-R型机器上运行测试程序时，偏移寻址结果分布如下，要求长度应满足≥85%的访问，偏移长度该如何选择？



选择结果：长度仅1种时，为12~16位(可结合指令字长再确定)；

长度有2种时，为6位、14位 ←利于缩短平均码长

第6节　指令格式设计

※主要内容：操作码设计、指令格式设计及优化

一、操作码设计

设计基础—已确定的操作类型、使用频率，指令字结构

设计内容—各条指令的操作码编码

1、操作码的设计

*设计步骤：确定需编码信息种类，确定编码方法，进行编码

*需编码信息种类：＝Σ(指令i操作类型数×指令i格式信息数)

＝操作功能×OPD类型种类→┘　　|

＝地址个数种类×目的OPD位置种类等→┘

思考①—整数加法指令支持8/16位OPD、单/双地址(目的OPD靠右)，编码个数?
栈操作指令支持16位的压栈/出栈操作，编码个数?

*编码方法：常采用扩展编码方式　←规整性较好、平均码长较短

思考②—3种频率操作码信息种数为4、15、56，编码时扩展目标？如何扩展？

思考①：4种，2种；　思考②：目标是平均码长最短，按频率分布扩展

2、操作码的编码

***扩展编码类型：** 方向有横向、纵向、混合，长度有等长、不等长

横向扩展		纵向扩展		混合扩展	
OP个数	扩展编码	OP个数	扩展编码	OP个数	扩展编码
15种	0000 … 1110	4种	0xx	10种	0000 … 1001
15种	1111 0000 … … 1111 1110	16种	1xx 0xx	20种	1010 00 … … 1110 11
15种	1111 1111 0000 … … … 1111 1111 1110	64种	1xx 1xx 0xx	64种	1111 00 0000 … … … 1111 11 1111
(OP频率分布1:1:1)		(OP频率分布1:4:16)		(OP频率分布1:2:5)	

不等长扩展

思考①—OP分布为4、15、56时，不等长横向扩展时如何编码？不等长混合扩展时如何编码？

思考①：3位、7位、13位；3位、6位、9位。

思考②—两个操作码分别为0100及01001001，编码正确吗？

***扩展方法：** 根据所有编码信息的频带进行扩展

思考③—各操作频率为0.12有4种、0.02有15种、0.004有55种，如何扩展？平均码长？

思考②：不正确，因为硬件不能识别0100后的1001是地址码还是操作码

思考③：4种—0XX，16种—1XX 0XX，64种—1XX 1XX 0XX；3b*4*0.12+6b*15*0.02+9b*55*0.004

二、指令格式设计及优化

设计基础—已确定的各指令功能、操作码、寻址方式

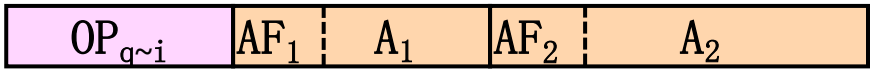
如：Ra←(Ra)+Rb或[c]，Ra←(Ra)+1

设计内容—各条指令的指令字格式

***指令格式的设计：**

方法—针对每条指令进行，组织地址码、合并操作码&地址码

第p种指令格式：



第k种指令格式：

注：①各指令的地址码中寻址方式位可单独编码(不要求所有指令相同)

②各指令的操作码可有多个编码(如OPD长度/目的OPD位置/寻址方式位)

③各指令的信息仅一种取值时可缺省编码(如OPD长度/AF，OP可隐含指明)

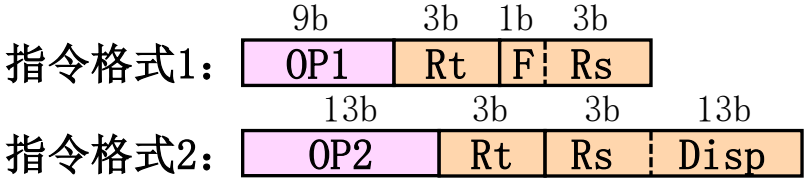
要求— ①操作码在前、地址码在后 ←利于正交性(如取指时取首字)

②地址码个数≤结构规定(否则隐式寻址/简化指令功能)

③指令字长＝*k**MEM编址单位 ←利于指令寻址

④预留部分OP编码 ←利于兼容性

例1—某ISA的MEM按字节编址、地址空间为32位，OPD可放在GPR、MEM中，指令中M_OPD个数≤1个，指令格式如下。



OP1/OP2—操作码，Rs/Rt—GPR号，
F—寻址方式位，　Disp—偏移量

(1)采用的是哪种ISA结构？显式OPD个数最多是多少？

(2)格式1中<F,Rs>支持哪些寻址方式？

(3)GPR个数、长度分别是多少？

(4)格式2中，某指令支持8/16位OPD，<Rs,Disp>支持基址寻址和变址寻址方式，目的OPD为Rt，该指令需占用几个操作码？

(5)格式1中，若去除F、将OP1改为10b，好处是什么？

答：(1)R-M型GPR结构，≤2个；　(2)REG寻址、REG间接寻址

(3)8个，32位；　(4)2种$_{OPD}$×2种$_{寻址方式}$＝4个

(5)增加了格式1指令数(如部分指令仅REG寻址)

例2—某CISC中，寄存器有8个32位GPR(可存储16位OPD)和8个32位FPR，存储器按字节编址、地址空间为32位，数据表示、寻址方式、ISA功能如下表所示，指令中显式OPD≤2个，请设计指令格式。

数据表示	指令名	功能	OPD寻址	op频率
定点(8/16/32b)	ADD/SUB	Rd←(Rd) op OPD, OPD=16/32b	,,①②	2种op*0.05
浮点(S/D)	INC等	Rd←(Rd) op 常数, OPD=32b	,,①常数	4种op*0.05
寻址方式	AND/OR等	Rd←(Rd) op OPD, OPD=16/32b	,,①②	6种op*0.002
①立即(8b)	LSL等移位	Rd←(Rd) op (R2), OPD=16/32b	,,②固定	4种op*0.002
②寄存器	FADD/FSUB等	Fd←(Fd) op (Fs), OPD=S	,,②	13种op*0.01
③寄存器间接	LD	Rd←OPD, OPD=8/16/32b	,③④	1种op*0.05
④基址(8b偏移)	ST	OPD←(Rd), OPD=8/16/32b	③④,	1种op*0.05

操作码设计—

需编码种类数： (操作类型+格式信息)

格式信息隐含表示(个数/位置)，寻址方式位放在地址码中；

0.05频率有2*2+4*1+1*3+1*3=14种，其余频率为13种、20种

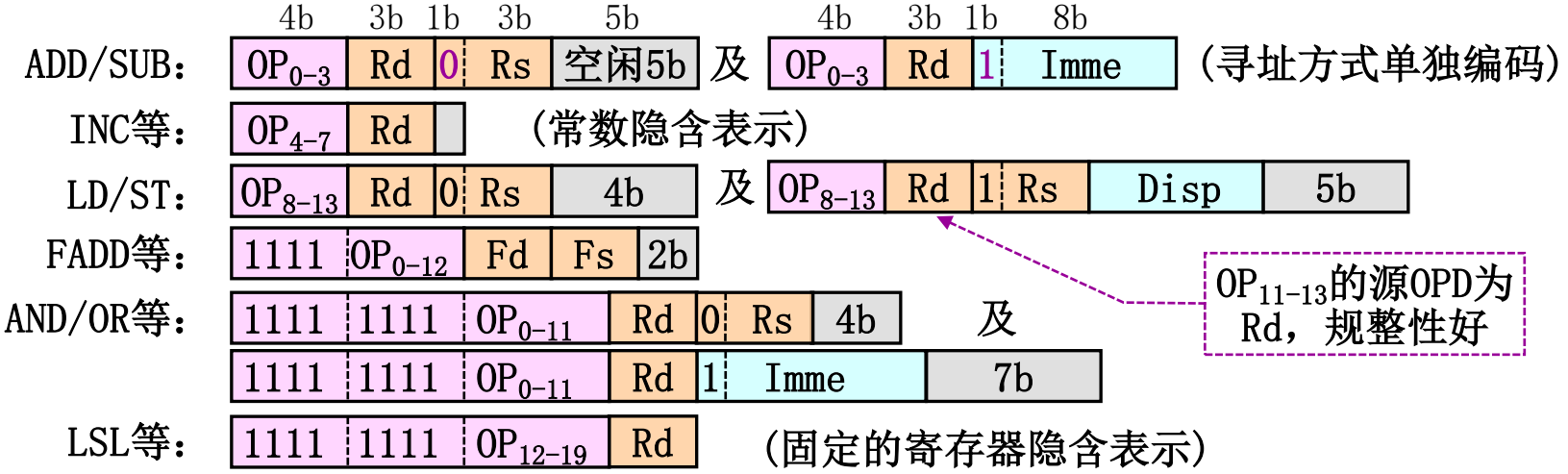
编码方法：

采用扩展编码(不等长横向扩展)，分别占4位、8位、13位

数据表示
定点(8/16/32b)
浮点(S/D)
寻址方式
①立即(8b)
②寄存器
③寄存器间接
④基址(8b偏移)

指令名	功能	OPD寻址	频率(/op)
ADD/SUB	Rd←(Rd) op OPD, OPD=16/32b	,,①②	2种op*0.05
INC等	Rd←(Rd) op 常数, OPD=32b	,,①常数	4种op*0.05
AND/OR等	Rd←(Rd) op OPD, OPD=16/32b	,,①②	6种op*0.002
LSL等移位	Rd←(Rd) op (R2), OPD=16/32b	,,②固定	4种op*0.002
FADD/FSUB等	Fd←(Fd) op (Fs), OPD=S	,,②	13种op*0.01
LD	Rd←OPD, OPD=8/16/32b	,③④	1种op*0.05
ST	OPD←(Rd), OPD=8/16/32b	③④,	1种op*0.05

指令格式设计— 字长=*k**8b，寻址方式位∈地址码，目的OPD靠左



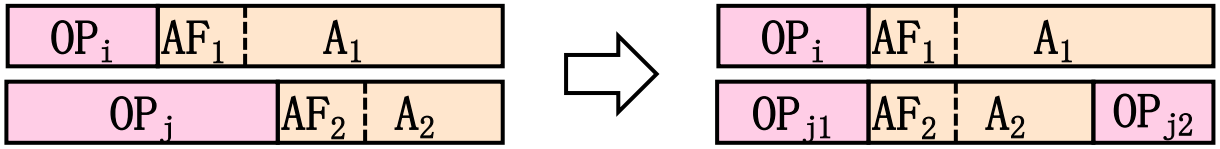
指令格式优化— INC、FADD等空闲位存放部分OP码； ←减少主OP码

AND/OR改为混合编码 ←缩短AND/OR指令字长

*指令格式的优化：归并格式，以优化性能(规整性/平均码长/执行速度)

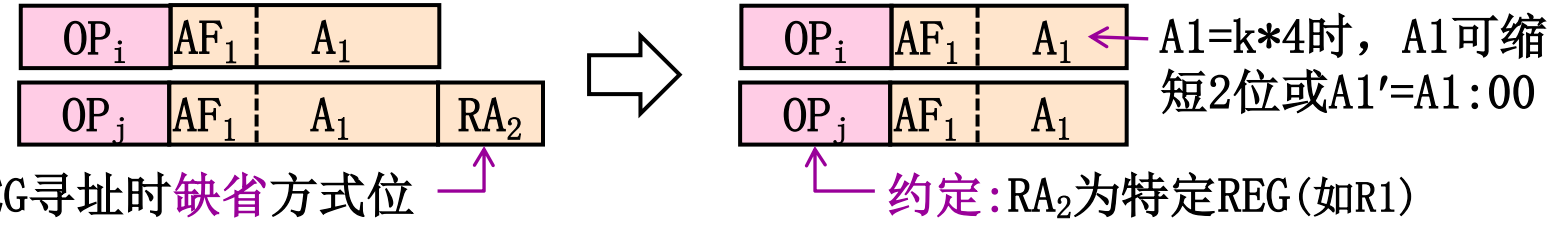
①操作码分开存放

←利于规整性



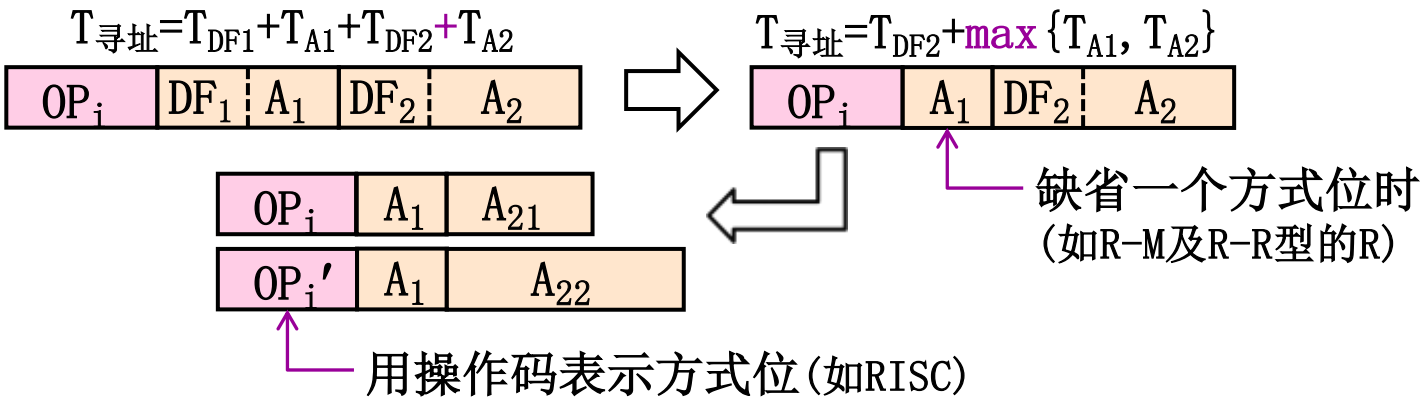
②隐含部分地址码

←利于规整性/码长、弊于编程



③隐含部分寻址方式位

←利于规整性、执行速度



三、MIPS指令系统分析

1、结构与功能

*ISA结构：R-R的GPR型，定长指令字(32b)，显式OPD≤3个

*数据表示：整数(s/u、8/16/32b)，浮点数(S/D)　←逻辑数∈整数

*指令功能：传送、算术/逻辑/移位、浮点、转移控制

└←区分定点/浮点(目的REG独立编址)

*寻址方式：

编址—MEM按字节编址、32位空间，大端、对齐方式存放；

REG按32位编址、5位空间(32个GPR)；

I/O与MEM统一编址

＝OPD长度，可≠数据表示

数据寻址—立即、REG、基址

指令寻址—相对(16b)、伪直接(26b)

2、指令格式

共3种，每条指令1~2种格式(寻址方式位∈op)，指令有30多条

	31~26(6b)	25~21(5b)	20~16(5b)	15~11(5b)	10~6(5b)	5~0(6b)
R-型指令	op	rs	rt	rd	shamt	func
I-型指令	op	rs	rt	imme/disp		
J-型指令	op	addr				

注：op及func—操作码，rs、rt及rd—寄存器号(源及目的OPD)，
shamt—移位位数，imme/disp—立即数或偏移量，addr—形式地址

***操作码的编码特征：**

采用扩展编码(op+func)，分开存放，

地址码个数、目的OPD位置隐式表示(通过op指明、靠右)

***地址码的编码特征：**

寻址方式位显式/隐式表示，地址码参数可分开存放

(使用op[下页]) (通过op指明) (如基址寻址[格式规整])

3、MIPS优化分析

R-型指令	000000	rs	rt	rd	00000	100000	rd←rs+rt，有符号加
	000000	rs	rt	rd	00000	100001	rd←rs+rt，无符号加
(000000)	000000	rs	00000	00000	00000	001000	PC←rs
I-型指令	001000	rs	rt	imme			rt←rs+imme，有符号加
(其余)	100011	rs	rt	disp			rt←M[(rs)＋disp]
	101011	rs	rt	disp			M[(rs)＋disp]←(rt)
	000100	rs	rt	disp			rs=rt时PC←(PC)+4+disp
J-型指令	000010	$(250)_{10}$					PC[27..0]←250<<2
(00001*)	000011	$(250)_{10}$					$31←PC+4,PC同上

***从代码效率方面优化：**

指令种类少(高频/通用)、功能简单(平衡)，

地址码个数多、长度短(代码效率高)

***从执行效率方面优化：**

指令格式少，地址码定长、寻址方式少； ←译码简单

操作数为R-R型(load/store指令除外) ←执行速度快

第二章 课后复习思考题

1、指令系统包含哪些内容？设计目标是什么？设计过程是什么？

2、ISA结构有哪些类型？与结构相关的指令格式参数有哪些？CISC/RISC风格对其有哪些影响？

3、若指令操作码为8位，存储器按字节编址、地址为64位，寄存器地址为6位，A、B、C、D为内存变量，针对堆栈型、累加器型、R-M风格GPR型、R-R风格GPR型指令系统，写出计算D=A+B-C的指令序列、代码大小。

4、数据表示与数据结构的关系？数据表示设计的内容、目标、方法？

5、指令系统功能设计的方法？CISC/RISC风格对功能设计及优化的影响？

6、数据存储部件可有哪些？确定编址单位的方法？寻址方式设计的内容、目标、方法？

7、x86及MIPS指令系统的基本特征、指令格式的优化方法？