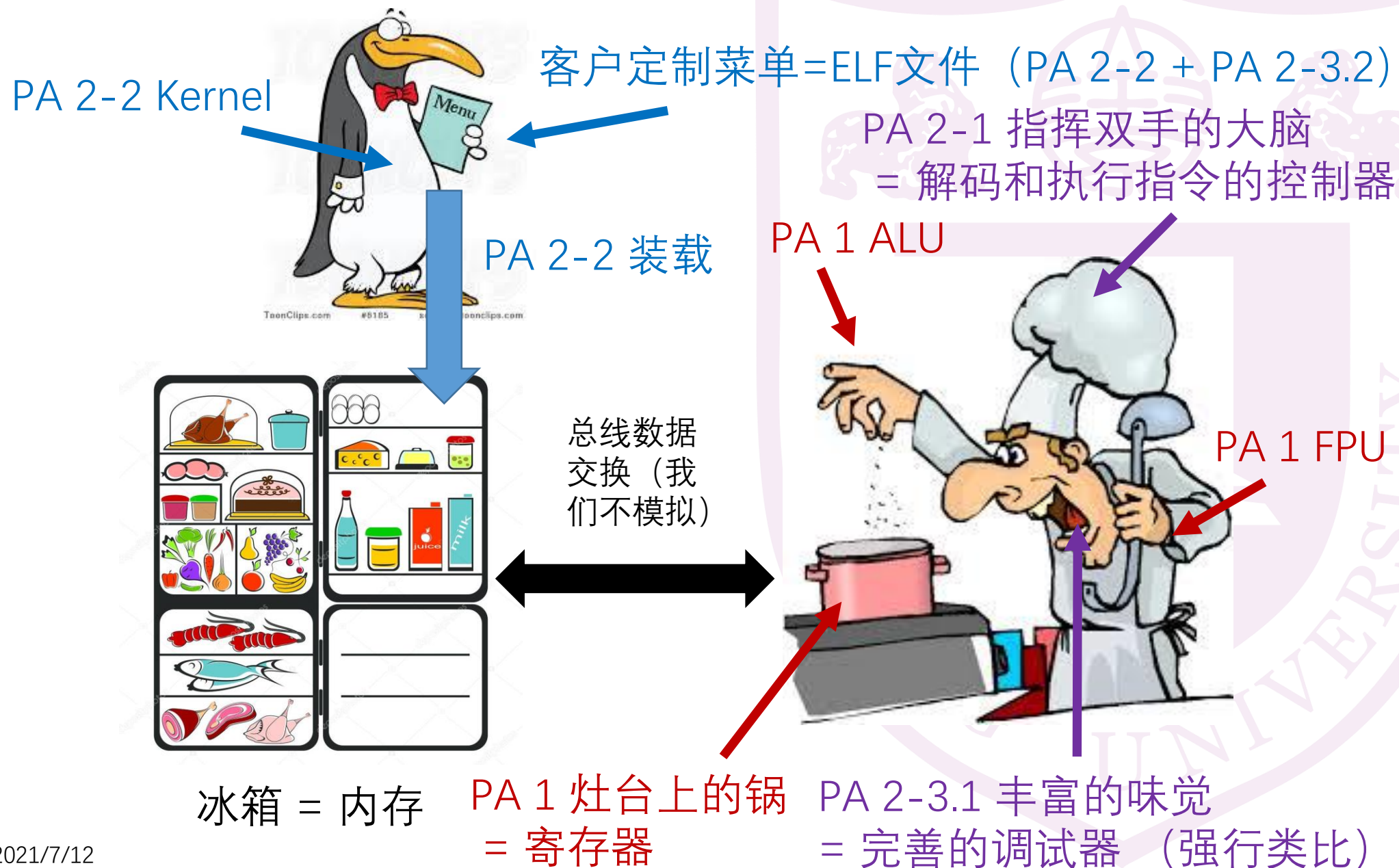


计算机系统基础
Programming Assignment

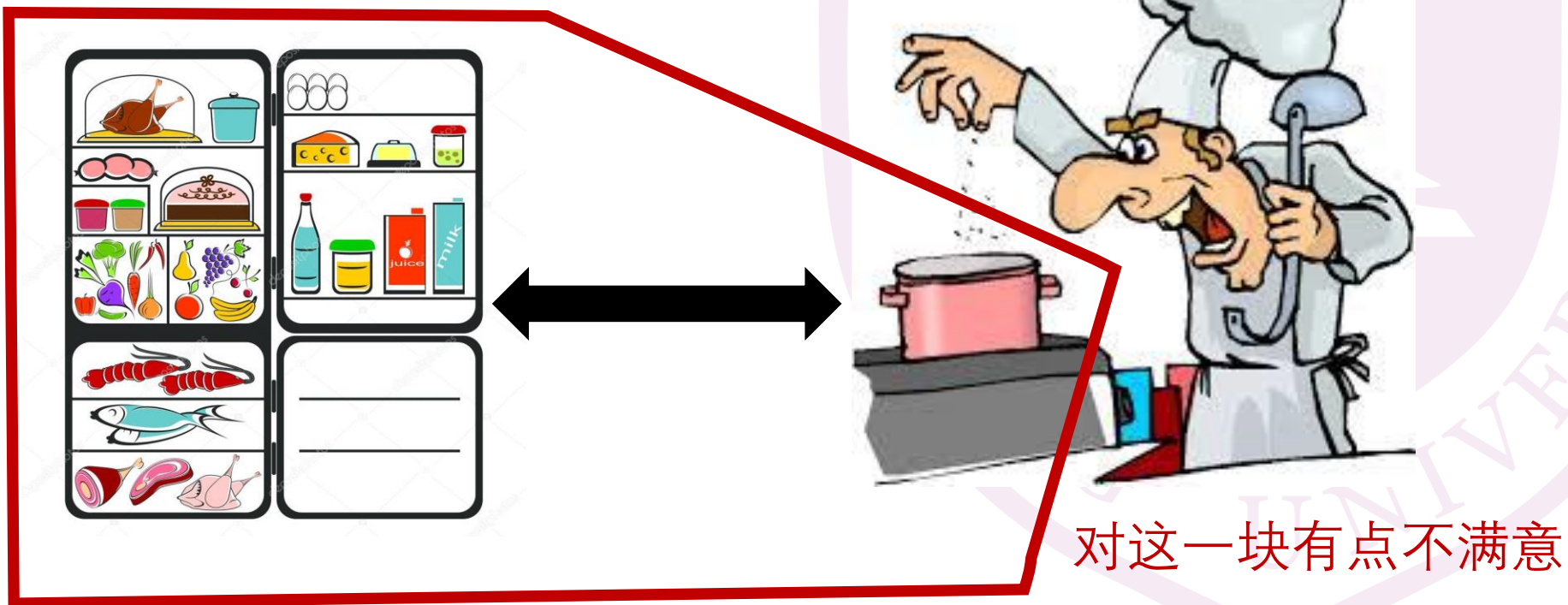
PA 3-1 Cache的模拟

2020年11月26日 / 11月27日
南京大学《计算机系统基础》课程组

前情提要 (以餐厅为类比)



PA 3的总体任务（以餐厅为类比）





PA 3-1 Cache的模拟

如何加快获取数据的速度？

PA 3-1 Cache的模拟

- 动机：频繁的访存大大影响了运算的速度
 - 每次取指令要访存
 - `instr_fetch()`
 - 每次取/存操作数要访存
 - `operand_read()` / `write()`
 - 调用 `vaddr_read()` / `write()`
 - 一条指令访存三四次，时间都花在访存上了



PA 3-1 Cache的模拟

- 怎么办？

- 灶台（CPU芯片）上还有点空间，我放些小碟子来临时存放马上要用的东西（数据）
- 这就是Cache，内存的Cache



后厨冰箱送出来，马上要烫的食材，在台面盘子里放好。要吃的时候直接盘子里夹。否则服务员一趟趟的跑，菜没吃几口，汤都烧干了。

内存里读出来，马上要用的数据，在Cache中摆好。要用的时候直接从Cache中读。否则每次都访存，指令没执行几条，时间都消耗了。

PA 3-1 Cache的模拟

- 基本原理：局部性

- 时间局部性

- 刚刚被访问过的数据很有可能再被访问（循环）
 - 刚刚被点过的菜很有可能再被点（食堂的炒饭，每次有个人点个什么，他都炒一大锅一样的，马上就有人要）

- 空间局部性

- 刚刚被访问数据附近的数据很有可能马上再被访问（数组、指令序列）
 - 刚点过牛肉很可能马上再点羊肉

PA 3-1 Cache的模拟

- Cache设计时的三大问题
 - 问题一： Cache行和主存块的映射
 - 什么菜放什么盘子？
 - 问题二： Cache中主存块的替换算法
 - 盘子摆满了怎么办？
 - 问题三： Cache一致性问题
 - 盘子里的菜被改了怎么办？（这个问题没法类比……）

PA 3-1 Cache的模拟

- 问题一： Cache行和主存块的映射

- 什么菜放什么盘子？

- 首先， 盘子取多大？

- 对主存和Cache都以相同的尺寸进行划分成单元

- 设，都以 2^b 字节为大小进行划分

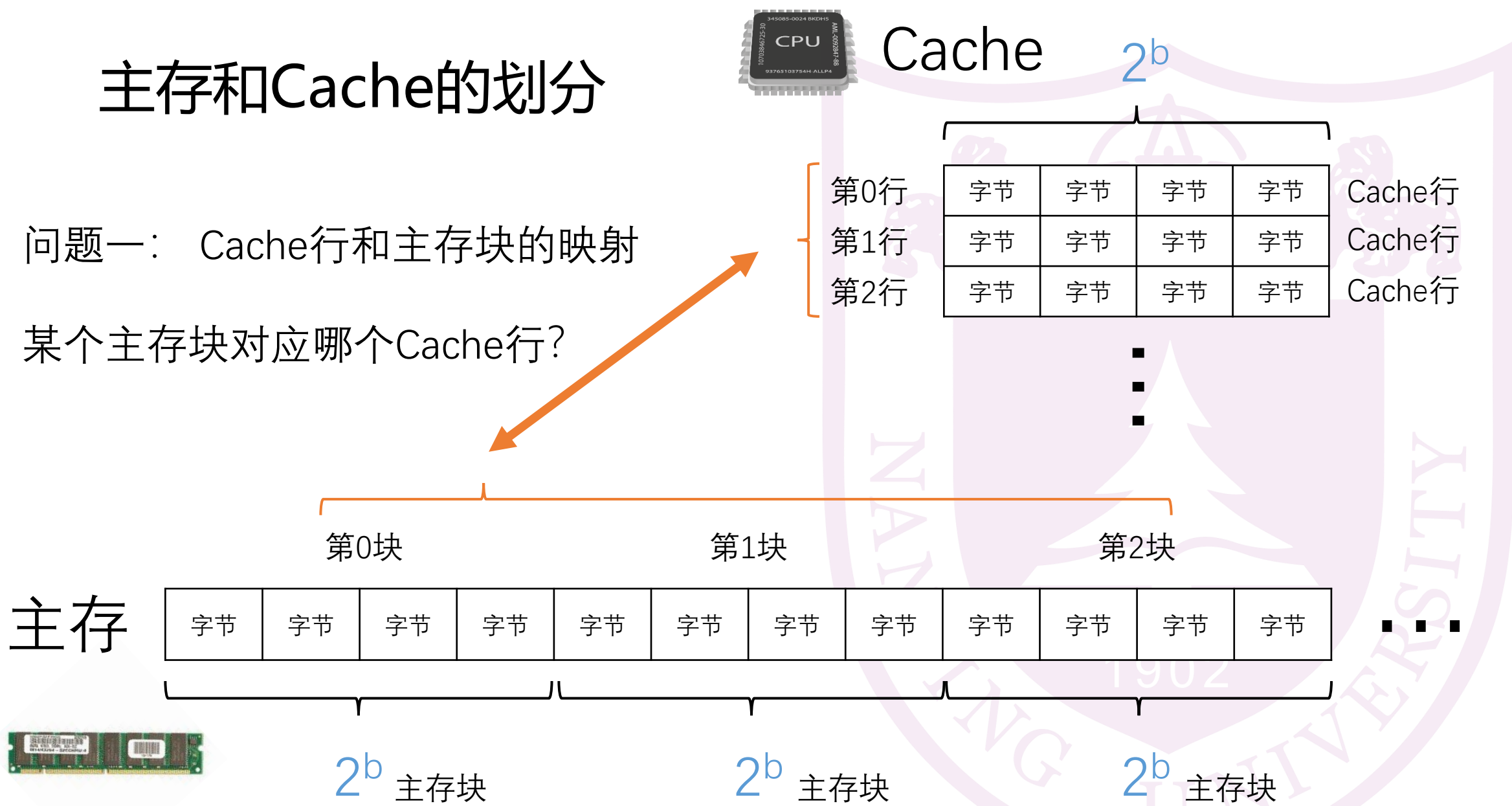
- 主存中的一个单元叫一个主存块 (block)

- Cache中的一个单元叫一个Cache行 (line) 或槽 (slot)

主存和Cache的划分

问题一： Cache行和主存块的映射

某个主存块对应哪个Cache行？



PA 3-1 Cache的模拟

- 问题一： Cache行和主存块的映射
 - 什么菜放什么盘子？
- 第二，主存块和Cache行怎么对应？
 - 直接映射
 - 一个萝卜一个盘~（坑→盘）
 - 全相联映射
 - 是盘就能占
 - 组相联映射
 - 划分成组，不能占别组的盘，但自己的组内是盘就能占

PA 3-1 Cache的模拟

- 问题一： Cache行和主存块的映射

- 直接映射法

- 都以 2^b 字节为大小进行划分

- 主存可以被划分成 2^m 个块

- Cache的数据区提供了 2^c 个行

- 如何一个萝卜一个盘？

- 模映射法：内存中第M个块的块号 $M \bmod \text{Cache的行数} \Rightarrow \text{Cache行号}$

- 对应到主存物理地址的划分？

PA 3-1 Cache的模拟

- 问题一： Cache行和主存块的映射

- 直接映射法

- 都以 2^b 字节为大小进行划分

- 主存可以被划分成 2^m 个块

- Cache的数据区提供了 2^c 个行

- 如何一个萝卜一个盘？

- 模映射法：内存中第M个块的块号 $M \bmod \text{Cache的行数} \Rightarrow \text{Cache行号}$

- 对应到主存物理地址的划分？

举例：

块大小 = 4B

主存分为 64 块

Cache数据区有 4 行

第0块 -> 第0行

第1块 -> 第1行

第2块 -> 第2行

第3块 -> 第3行

第4块 -> 第0行

...

第63块 -> 第3行

PA 3-1 Cache的模拟

- 问题一： Cache行和主存块的映射

- 直接映射法

- 都以 2^b 字节为大小进行划分

- 主存可以被划分成 2^m 个块

- Cache的数据区提供了 2^c 个行

- 如何一个萝卜一个盘？

- 模映射法：内存中第M个块的块号 $M \bmod \text{Cache的行数} \Rightarrow \text{Cache行号}$

- 对应到主存物理地址的划分？

给定NEMU中的一个32位的物理地址，如何计算其块号？
如何计算其行号？

块号 = 物理地址 / 2^b (取整)
行号 = 块号 $\bmod 2^c$

PA 3-1 Cache的模拟

- 问题一： Cache行和主存块的映射

- 直接映射法

- 都以 2^b 字节为大小进行划分

- 主存可以被划分成 2^m 个块

- Cache的数据区提供了 2^c 个行

- 如何一个萝卜一个盘？

- 模映射法：内存中第M个块的块号 $M \bmod \text{Cache的行数} \Rightarrow \text{Cache行号}$

- 对应到主存物理地址的划分？

给定NEMU中的一个32位的物理地址，**如何计算其块号？**
如何计算其行号？

块号 = 物理地址 / 2^b (取整)
行号 = 块号 $\bmod 2^c$

PA 3-1 Cache的模拟

- 问题一： Cache行和主存块的映射

- 直接映射法

- 都以 2^b 字节为大小进行划分

- 主存可以被划分成 2^m 个块

- Cache的数据区提供了 2^c 个行

- 如何一个萝卜一个盘？

- 模映射法：内存中第M个块的块号 $M \bmod \text{Cache的行数} \Rightarrow \text{Cache行号}$

- 对应到主存物理地址的划分？

给定NEMU中的一个32位的物理地址，**如何计算其块号？**
如何计算其行号？

块号 = 物理地址 / 2^b (取整)

行号 = 块号 $\bmod 2^c$



$m + b = 32$ 位物理地址

PA 3-1 Cache的模拟

- 问题一： Cache行和主存块的映射

- 直接映射法

- 都以 2^b 字节为大小进行划分

- 主存可以被划分成 2^m 个块

- Cache的数据区提供了 2^c 个行

- 如何一个萝卜一个盘？

- 模映射法：内存中第M个块的块号 $M \bmod \text{Cache的行数} \Rightarrow \text{Cache行号}$

- 对应到主存物理地址的划分？

给定NEMU中的一个32位的物理地址，如何计算其块号？

如何计算其行号？

块号 = 物理地址 / 2^b (取整)

行号 = 块号 $\bmod 2^c$



$m + b = 32$ 位物理地址

PA 3-1 Cache的模拟

- 问题一：Cache行和主存块的映射

- 直接映射法

- 都以 2^b 字节为大小进行划分

- 主存可以被划分成 2^m 个块

- Cache的数据区提供了 2^c 个行

- 如何一个萝卜一个盘？

- 模映射法：内存中第M个块的块号 $M \bmod \text{Cache的行数} \Rightarrow \text{Cache行号}$

- 对应到主存物理地址的划分？

给定NEMU中的一个32位的物理地址，如何计算其块号？
如何计算其行号？

块号 = 物理地址 / 2^b (取整)

行号 = 块号 $\bmod 2^c$



PA 3-1 Cache的模拟

- 问题一：Cache行和主存块的映射

- 直接映射法
- Cache的组织

主存（物理）地址：



Cache行号（就是Cache行构成的数组之下标，不用显式给了）

Cache:

0	有效位	标记	2 ^b 字节块（行）内数据
1	有效位	标记	2 ^b 字节块（行）内数据
2	有效位	标记	2 ^b 字节块（行）内数据
3	有效位	标记	2 ^b 字节块（行）内数据

额外的1比特有效位：为0表示Cache行无效，为1表示有效

PA 3-1 Cache的模拟

- 问题一： Cache行和主存块的映射
 - 直接映射法
 - 所以说我有了主存物理地址，我想读数据，怎么知道它对应Cache里的哪一行？在不在Cache里？

主存（物理）
地址：



Cache: 0
1
2
3

有效位	标记	2^b 字节块（行）内数据
有效位	标记	2^b 字节块（行）内数据
有效位	标记	2^b 字节块（行）内数据
有效位	标记	2^b 字节块（行）内数据

PA 3-1 Cache的模拟

- 问题一： Cache行和主存块的映射
 - 直接映射法
 - 所以说我有了主存物理地址，我想读数据，怎么知道它对应Cache里的哪一行？在不在Cache里？

第一步：找行号

主存（物理）
地址：



Cache: 0
1
2
3

有效位	标记	2^b 字节块（行）内数据
有效位	标记	2^b 字节块（行）内数据
有效位	标记	2^b 字节块（行）内数据
有效位	标记	2^b 字节块（行）内数据

PA 3-1 Cache的模拟

- 问题一： Cache行和主存块的映射
 - 直接映射法
 - 所以说我有了主存物理地址，我想读数据，怎么知道它对应Cache里的哪一行？在不在Cache里？

主存（物理）
地址：



第一步：找行号

第二步：比标记

- 相等？ 有希望！ 继续
- 不等？ 没希望！ 读内存块去

Cache: 0
1
2
3

有效位	标记	2 ^b 字节块（行）内数据
有效位	标记	2 ^b 字节块（行）内数据
有效位	标记	2 ^b 字节块（行）内数据
有效位	标记	2 ^b 字节块（行）内数据

PA 3-1 Cache的模拟

- 问题一： Cache行和主存块的映射
 - 直接映射法
 - 所以说我有了主存物理地址，我想读数据，怎么知道它对应Cache里的哪一行？在不在Cache里？

主存（物理）
地址：



第一步：找行号

第二步：比标记

- 相等？有希望！继续
- 不等？没希望！读内存块去

第三步：看有效位

- 为1？中！读Cache
- 为0？没中！读内存块去

Cache: 0
1
2
3

有效位	标记	2^b 字节块（行）内数据
有效位	标记	2^b 字节块（行）内数据
有效位	标记	2^b 字节块（行）内数据
有效位	标记	2^b 字节块（行）内数据

PA 3-1 Cache的模拟

- 问题一： Cache行和主存块的映射
 - 直接映射法
 - 所以说我有了主存物理地址，我想读数据，怎么知道它对应Cache里的哪一行？在不在Cache里？

主存（物理）
地址：



第一步：找行号

第二步：比标记

- 相等？有希望！继续
- 不等？没希望！读内存块去

第三步：看有效位

- 为1？中！读cache
- 为0？没中！读内存块去

Cache: 0
1
2
3

有效位	标记	2 ^b 字节块（行）内数据
有效位	标记	2 ^b 字节块（行）内数据
有效位	标记	2 ^b 字节块（行）内数据
有效位	标记	2 ^b 字节块（行）内数据

第四步：如命中，按照块内地址读cache

第四步*：如不命中，按照物理内存读主存，把块搬入cache，填好标记和有效位，按照块内地址读cache

举例

块大小： 4B

物理地址： 32位

Cache行数： 4行

直接映射法

主存物理地址（读1字节）：

00...00101001

Cache:

		数据区			
		0	1	2	3
0	1 00	7D	3A	4C	56
1	1 10	FF	01	83	9D
2	1 11	33	8D	9C	7F
3	1 01	BC	A5	34	21

标记（高位0省略）

举例

块大小：4B

物理地址：32位

Cache行数：4行

直接映射法

2位块内偏移量

主存物理地址（读1字节）：

00...00101001

块号

块内偏移量

Cache:

0
1
2
3

有效位

数据区

		0	1	2	3
0	1 00	7D	3A	4C	56
1	1 10	FF	01	83	9D
2	1 11	33	8D	9C	7F
3	1 01	BC	A5	34	21

标记（高位0省略）

举例

块大小：4B

物理地址：32位

Cache行数：4行

直接映射法

主存物理地址（读1字节）：

00...00101001

2位行号

标记

行号

块内偏移量

Cache:

0
1
2
3

有效位

数据区

		0	1	2	3
0	1 00	7D	3A	4C	56
1	1 10	FF	01	83	9D
2	1 11	33	8D	9C	7F
3	1 01	BC	A5	34	21

标记（高位0省略）

举例

块大小：4B

物理地址：32位

Cache行数：4行

直接映射法

主存物理地址（读1字节）：

00...00101001

标记 行号 块内偏移量

Cache:

0
1
2
3

有效位

数据区

		0	1	2	3
0	1 00	7D	3A	4C	56
1	1 10	FF	01	83	9D
2	1 11	33	8D	9C	7F
3	1 01	BC	A5	34	21

标记（高位0省略）

举例

块大小：4B

物理地址：32位

Cache行数：4行

直接映射法

主存物理地址（读1字节）：

00...00101001

标记 行号 块内偏移量

Cache:

有效位为1

0
1
2
3

有效位

		数据区			
		0	1	2	3
0	1 00	7D	3A	4C	56
1	1 10	FF	01	83	9D
2	1 11	33	8D	9C	7F
3	1 01	BC	A5	34	21

标记（高位0省略）

举例

块大小：4B

物理地址：32位

Cache行数：4行

直接映射法

主存物理地址（读1字节）：

00...00101001

标记 行号 块内偏移量

标记不同，
Cache缺失

Cache:

	有效位		0	1	2	3
0	1	00	7D	5A	4C	56
1	1	10	FF	01	83	9D
2	1	11	33	8D	9C	7F
3	1	01	BC	A5	34	21

标记（高位0省略）

举例

块大小：4B

物理地址：32位

Cache行数：4行

直接映射法

主存物理地址（读1字节）：

00...00101001

标记 行号 块内偏移量

块号

将第00..001010主存块读入Cache，替换当前行，更新有效位和标记

Cache:

0	1	00	7D	3A	4C	56
1	1	10	FF	01	83	9D
2	1	10	51	99	07	85
3	1	01	BC	A5	34	21

标记（高位0省略）

举例

块大小：4B

物理地址：32位

Cache行数：4行

直接映射法

主存物理地址（读1字节）：

00...00101001

标记 行号 块内偏移量

Cache:

0	1	00	7D	3A	4C	56
1	1	10	FF	01	83	9D
2	1	10	51	99	07	85
3	1	01	BC	A5	34	21

标记（高位0省略）

读Cache
得到99

举例

块大小：4B

物理地址：32位

Cache行数：4行

直接映射法

主存物理地址（读1字节）：

00...00101010

标记 行号 块内偏移量

Cache:

有效位		数据区			
		0	1	2	3
0	1 00	7D	3A	4C	56
1	1 10	FF	01	83	9D
2	1 10	51	99	07	85
3	1 01	BC	A5	34	21

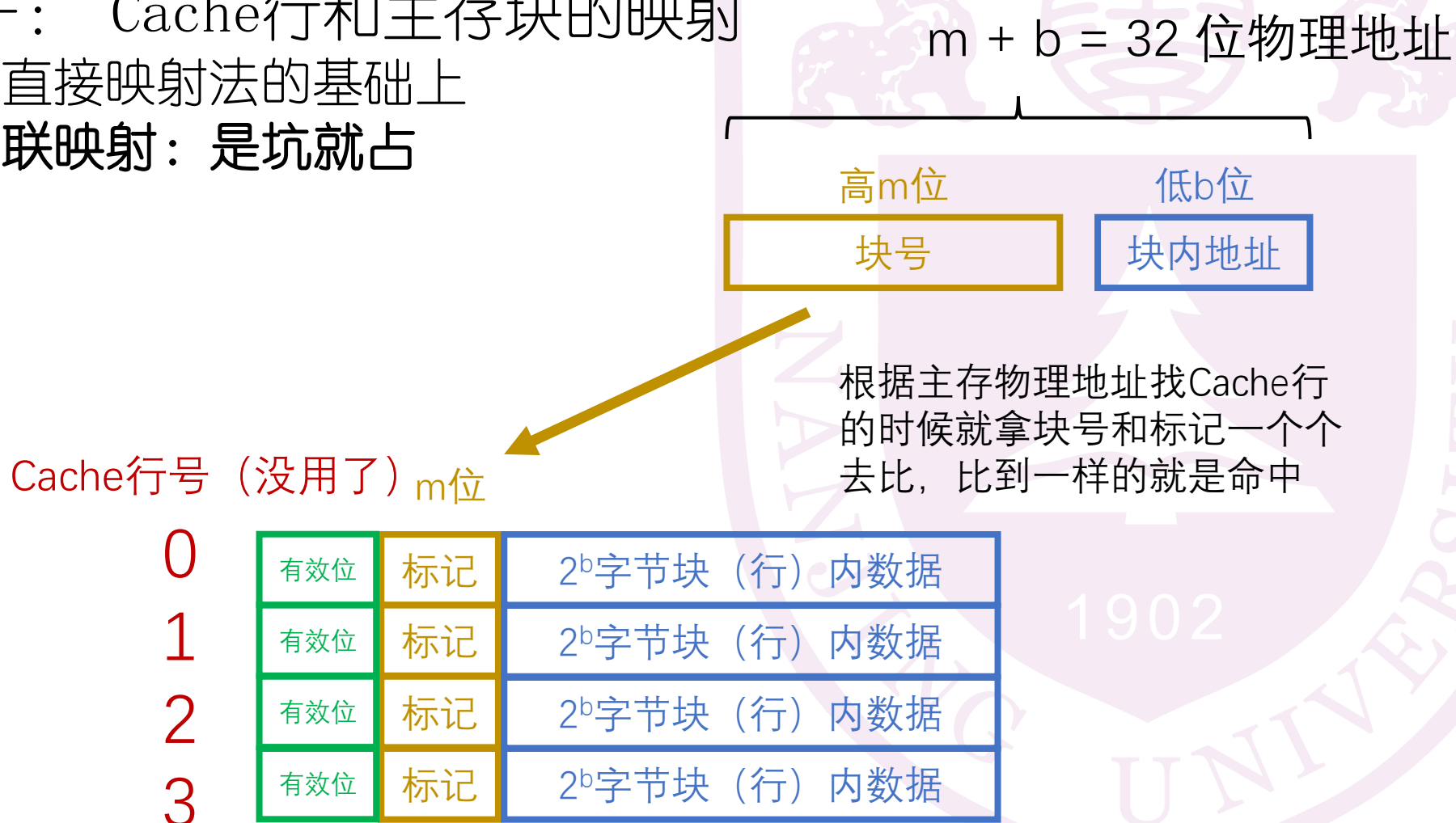
标记（高位0省略）

Cache命中，读
Cache得到07

PA 3-1 Cache的模拟

- 问题一：Cache行和主存块的映射

- 理解直接映射法的基础上
- 全相联映射：是坑就占



PA 3-1 Cache的模拟

- 问题一： Cache行和主存块的映射
 - 直接映射法的基础上
 - 组相联映射：折中方案
 - Cache的数据区提供了 2^c 个行
 - 划分成 2^q 个组（ q 肯定小于 c ）
 - 组间采用直接映射
 - 组内采用全相联映射
 - 一个组里有 $2^s = 2^{c-q}$ 行，称为 2^s 路组相联

PA 3-1 Cache的模拟

- 问题一： Cache行和主存块的映射
 - 直接映射法的基础上
 - 组相联映射： 折中方案
 - Cache的数据区提供了 2^c 个行
 - 划分成 2^q 个组 (q肯定小于c)
 - 一个组里有 $2^s = 2^{c-q}$ 行，称为 2^s 路组相联

主存（物理）地址：



PA 3-1 Cache的模拟

- 问题一： Cache行和主存块的映射
 - 直接映射法的基础上
 - 组相联映射： 折中方案
 - 有了物理地址找到Cache行的过程
 - 第一步： 根据组号找到相应的组
 - 第二步： 组内的Cache行一个个比标记， 命中？不命中？
 - 第三步： 看有效位， 命中？不命中？
 - 第四步： 命中？读Cache： 不命中？先搬主存块到Cache， 再读Cache

主存（物理）地址：

Cache组织（参见课本）



PA 3-1 Cache的模拟

- 问题一： Cache行和主存块的映射
 - 什么菜放什么盘子？
- 第二，主存块和Cache行怎么对应？
 - 直接映射
 - 一个萝卜一个盘~（坑→盘）
 - 全相联映射
 - 是盘就能占
 - 组相联映射
 - 划分成组，不能占别组的盘，但自己的组内是盘就能占

各有优劣

PA 3-1 Cache的模拟

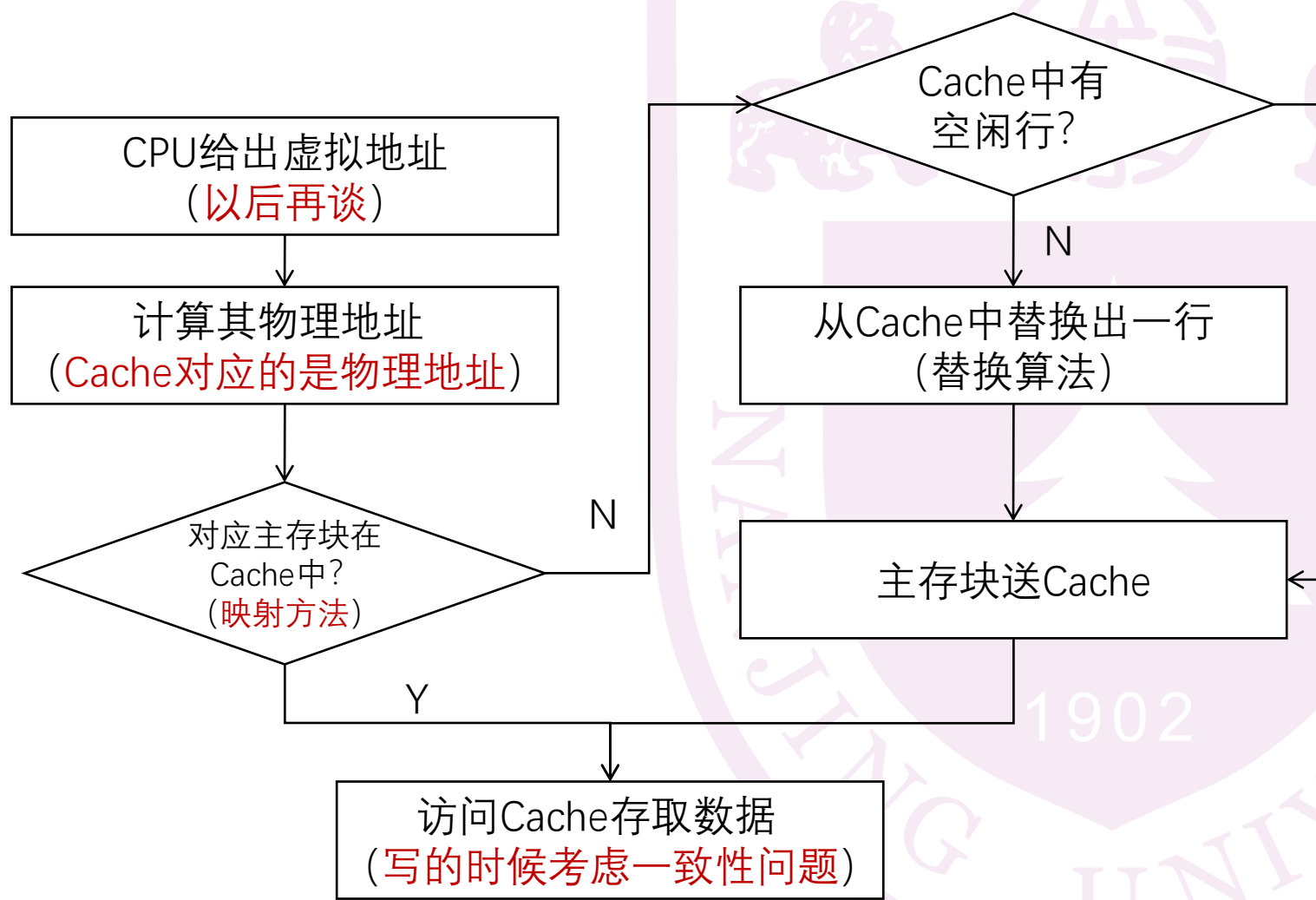
- 问题二： Cache中主存块的替换算法
 - 盘子摆满了怎么办？
- 根据程序特性各有优劣（略，参见课本）
 - 先进先出法
 - 最近最少用法
 - 最近不经常用法
 - 随机替换算法



PA 3-1 Cache的模拟

- 问题三： Cache一致性问题
 - 发生于Cache被写了之后，Cache和主存内容不一致了
- 通常两种策略（略，参见课本）
 - 全写法（write through）：写Cache同时写主存
 - 写分配法（write allocate）：写缺失会导致Cache装载
 - 非写分配法（not write allocate）：写缺失不导致Cache装载
 - 回写法（write back）：Cache行被替换时再写主存块
 - 需要额外设置脏位

CPU访问Cache的过程



PA 3-1 Cache的模拟（要求）

- 在NEMU中实现一个cache，它的性质如下：

1. cache block存储空间的大小为64B
2. cache存储空间的大小为64KB
3. 8-way set associative
4. 标志位只需要valid bit即可
5. 替换算法采用随机方式
6. write through
7. not write allocate

PA 3-1 Cache的模拟 (NEMU代码)

- 第一步：编辑include/config.h

```
#define CACHE_ENABLED
```

PA 3-1 Cache的模拟 (NEMU代码)

- 第二步：编辑 `nemu/include/memory/mmu/cache.h`
 - 在里面定义一个Cache行所对应的结构体，假设名称为 `CacheLine`
 - 结构的几个要点
 - 标志位有哪些？（只需要valid bit即可）
 - 标记部分多少位？（8-way set associative, cache block存储空间的大小为64B）
 - 数据区怎么表示？（cache block存储空间的大小为64B）

PA 3-1 Cache的模拟 (NEMU代码)

- 第三步：编辑 `nemu/src/memory/mmu/cache.c`

- 在里面为模拟Cache分配存储空间，也就是在`cache.c`中定义一个全局变量，其形式为在`cache.h`中定义的`CacheLine`类型的数组
- 因为是用内存中的数组模拟的`cache`，所以完成模拟后性能没法提升
- 该数组的要点
 - Cache该有多少行？
(`cache block`存储空间的大小为64B，`cache`存储空间的大小为64KB)

PA 3-1 Cache的模拟 (NEMU代码)

• 第四步：编辑 `nemu/include/memory/mmu/cache.h`

- 在里面声明一些cache所提供的函数

- `init_cache()`;

- 初始化cache，核心就是把valid bit都清0

- `uint32_t cache_read(paddr_t paddr, size_t len);`

- 读cache

- 前两个参数分别是物理地址和读的字节数

- 最后一个参数就是cache数组的首地址，假设在cache.h中定义的Cache行的结构体名称为CacheLine

- 返回值为读出的数据

- `void cache_write(paddr_t paddr, size_t len, uint32_t data);`

- 写cache

- 除data参数是待写的数据外，其它参数含义和cache_read()相同

- 不需要返回值

PA 3-1 Cache的模拟 (NEMU代码)

• 第五步：编辑 `nemu/src/memory/mmu/cache.c`

• 实现在`cache.h`中所声明的函数

- `init_cache();`
 - 初始化cache, 核心就是把valid bit都清0
- `uint32_t cache_read(paddr_t paddr, size_t len);`
 - 读cache
 - 根据paddr找到标记、组号、与块内地址
 - 根据组号去定位相应的组 (起始Cache行号)
 - 与组内的CacheLine比较标记和有效位
 - 命中怎样? 不命中 (缺失) 又怎样?
 - 组满了怎办? (随机替换算法)
- `void cache_write(paddr_t paddr, size_t len, uint32_t data);`
 - 写cache
 - 和`cache_read()`采用同样过程根据paddr定位CacheLine
 - 命中怎样? 不命中 (缺失) 又怎样? (`not write allocate`)
 - 写cache, 同时主存里面对应的块怎么办? (`write through`)

注意数据跨cache行的情形!

PA 3-1 Cache的模拟 (NEMU代码)

•第六步：编辑 `nemu/src/memory/memory.c`

- `#include "memory/mmu/cache.h"`
- 在 `init_mem()` 函数中调用 `init_cache()`
 - 加入条件编译选项以便日后通过修改 `include/config.h` 来跳过cache相关代码
 - 具体仿照教程中（或后页）针对 `paddr_read()` 和 `paddr_write()` 的修改方法
- 在 `paddr_read()` 和 `paddr_write()` 中分别通过 `cache_read()` 和 `cache_write()` 函数来实现对物理地址的读写
- 可以设置一个简单的模拟计时器，如果cache命中则时间+10，cache缺失则时间+100，比较一下模拟的时间消耗有什么变化
 - 也可以加入对cache命中率的统计

PA 3-1 Cache的模拟 (NEMU代码)

```
uint32_t paddr_read(paddr_t paddr, size_t len) {  
    uint32_t ret = 0;  
    #ifdef CACHE_ENABLED  
        ret = cache_read(paddr, len, &L1_dcache);  
    #else  
        ret = hw_mem_read(paddr, len);  
    #endif  
    return ret;  
}
```

```
void paddr_write(paddr_t paddr, size_t len, uint32_t data) {  
    #ifdef CACHE_ENABLED  
        cache_write(paddr, len, data, &L1_dcache);  
    #else  
        hw_mem_write(paddr, len, data);  
    #endif  
}
```

nemu/src/memory/memory.c

提交时间

- PA 3-1 截止时间
 - 待定





祝大家学习快乐，身心健康！

欢迎大家踊跃参加问卷调查