嵌入式系统设计与应用

第二章 ARM处理器组成结构（1）

西安交通大学电信学院
孙宏滨

Hongbin Sun
Internal Teaching Use
Xi'an Jiaotong University

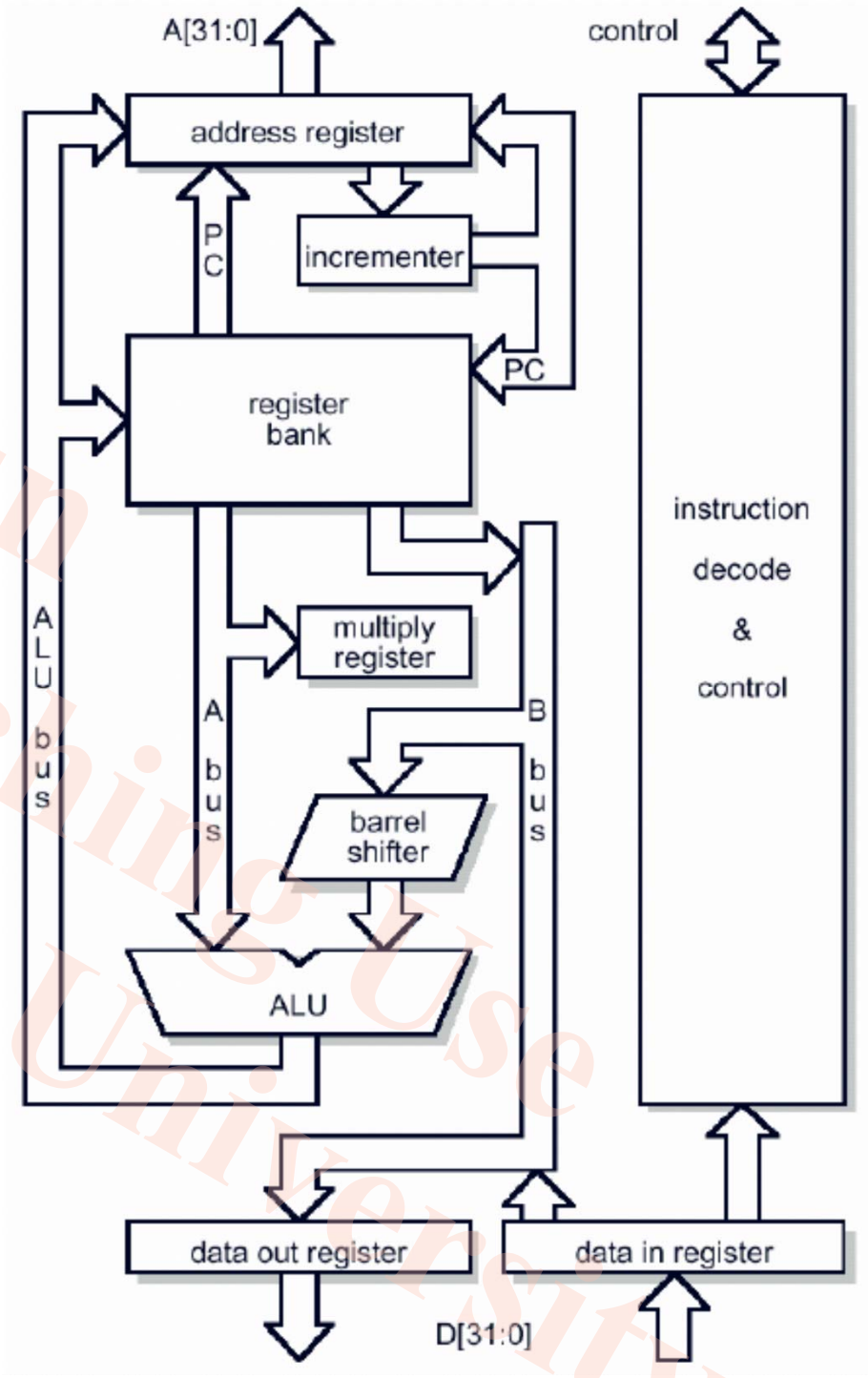
1 ARM处理器内核结构

- 83~85年：第一款ARM处理器采用3微米工艺；
- 90~95年：ARM6/ARM7，采用三级流水线结构；
- 95~02年：ARM9，采用五级流水线，程序/数据存储器分开；
- 02~12年：ARM Cortex系列处理器，ARMv7-A结构，ARM Cortex-A7,A8（单核），ARM Cortex-A9,A15（多核）
- 12年底：ARM Cotex-A50系列，64位ARMv8结构，进军PC和服务器市场。
- 本课程讲解内容主要基于ARM7.

Hongbin Sun
Internal Teaching Use
Xi'an Jiaotong University

ARM内部组成

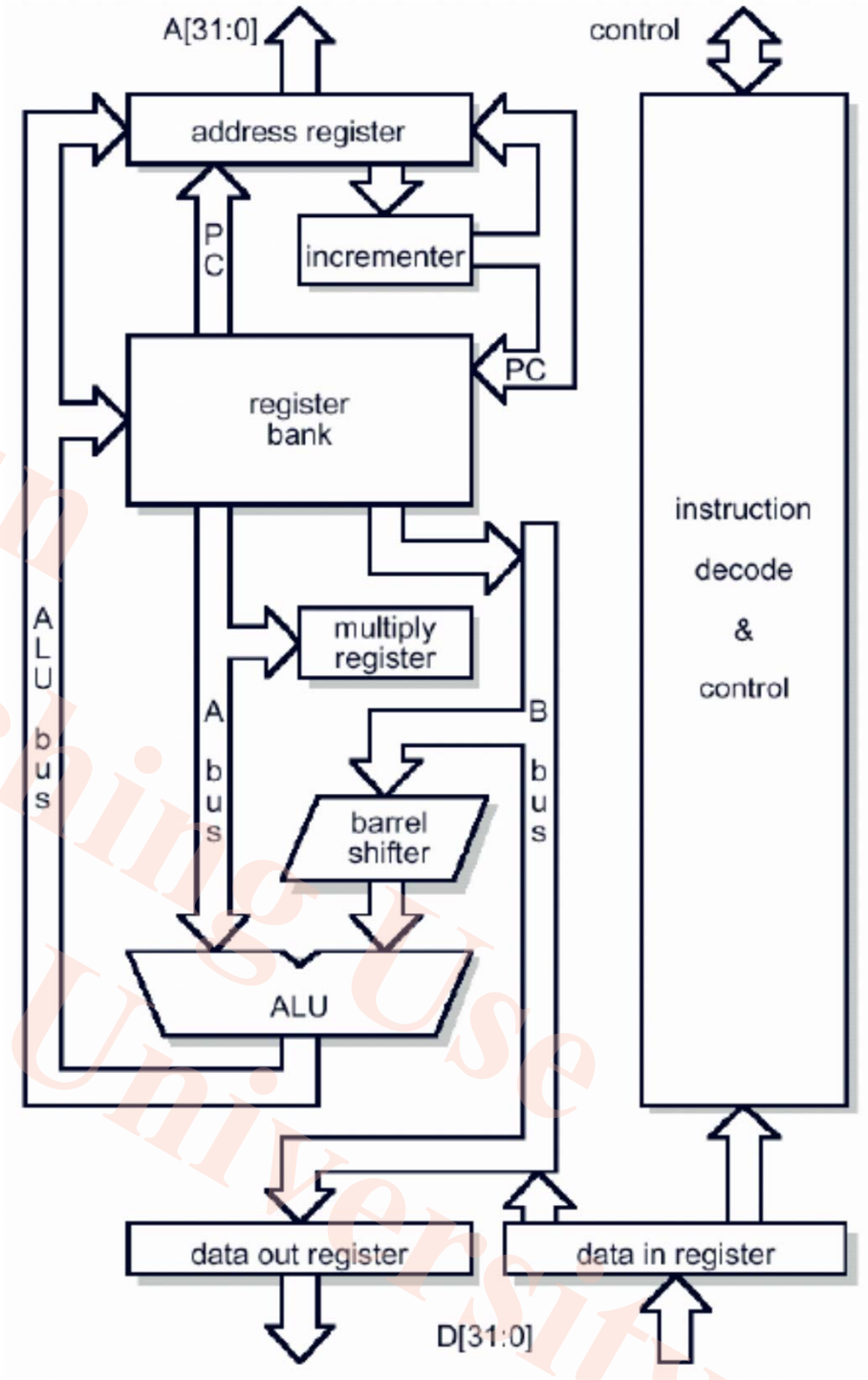
- 两个主要模块：数据通路（**data path**）与解码器（**decoder**）。
- 寄存器堆（**r0~r15**）
  - 两个读端口**A-bus/B-bus**
  - 一个写端口**ALU-bus**
  - 额外的对**r15**读写的端口
- 桶形移位寄存器：对第二操作数移位或循环移位若干数据位。
- **ALU**执行算术逻辑运算。



Hongbin Sun
International Teaching Use
Xi'an Jiaotong University

ARM内部组成(续)

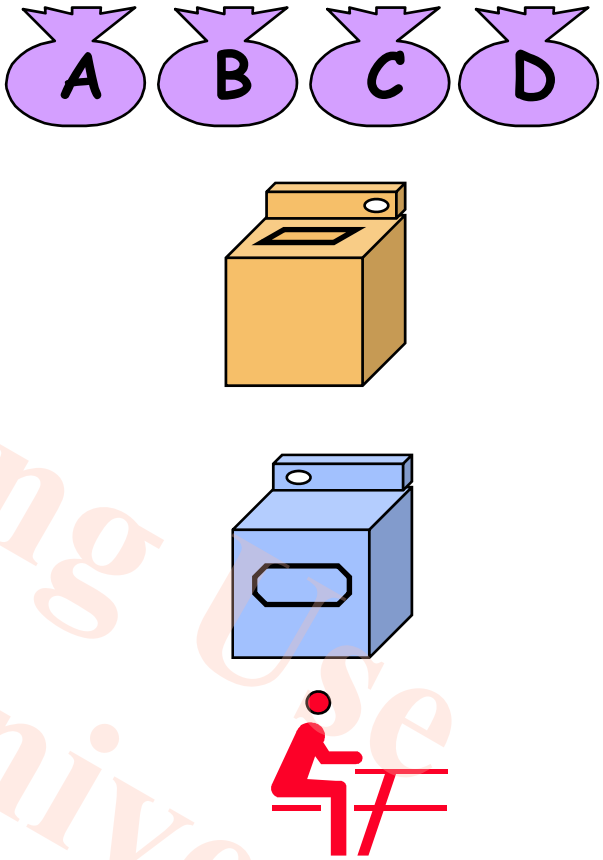
- 专门的PC累加器
- 地址寄存器：从ALU或PC
- 数据寄存器：保存对存储器写或读的数据
- 指令译码器对指令的机器码译码，产生数据通路所需的控制信号。
- 数据处理指令需要1个时钟周期，数据由A-bus&B-bus读，结果经ALU-bus写回寄存器堆。



Hongbin Sun Internal Teaching Use Xi'an Jiaotong University

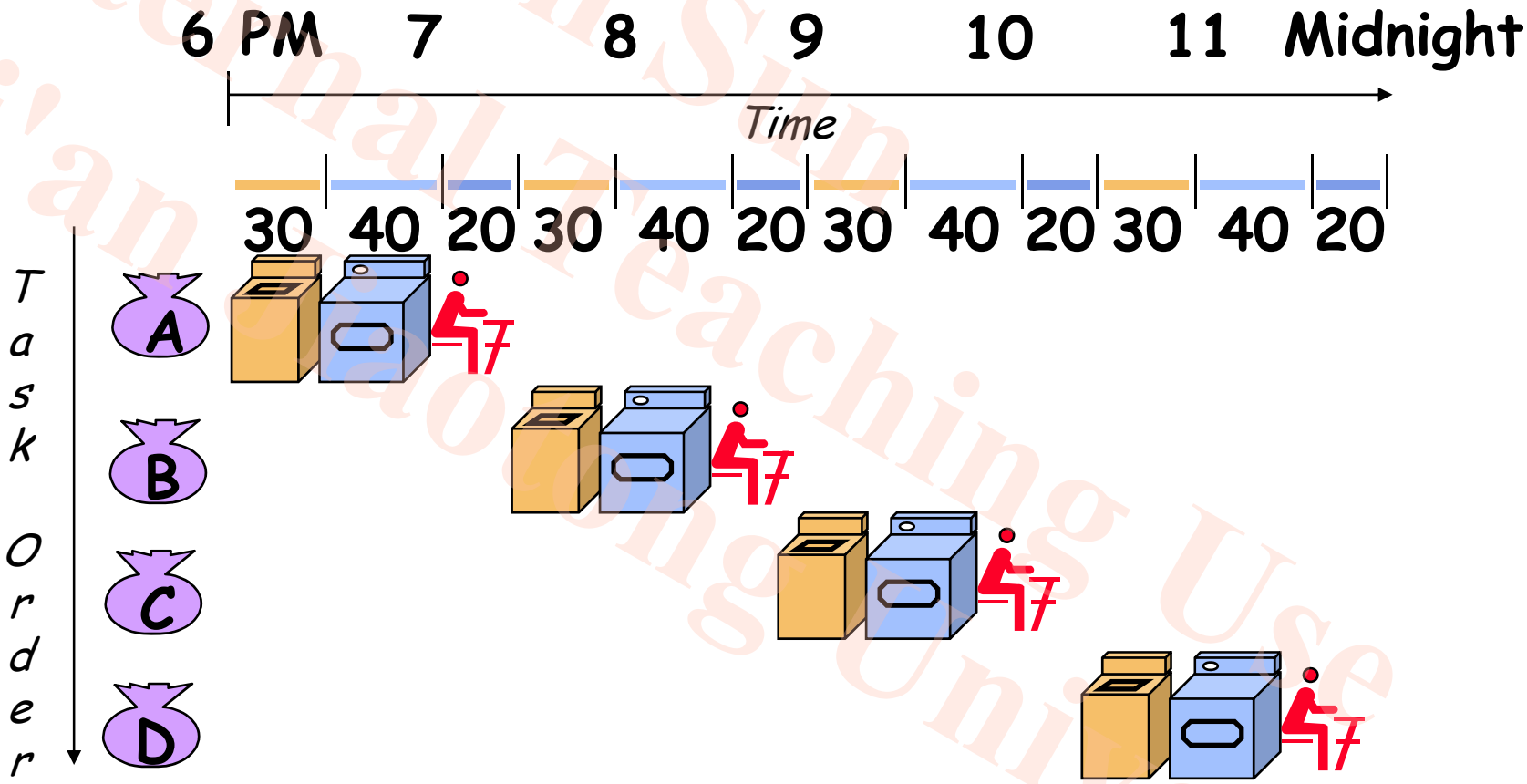
流水线的概念（1）

- 以洗衣服为例
- 假设有**Ann, Brian, Cathy, Dave**四个人，每个人都需要洗衣、烘干和叠衣服三个步骤。
  - 洗衣需要**30**分钟
  - 烘干需要**40**分钟
  - 叠衣服需要**20**分钟



Hongbin Shen
Internal Teaching Use
Xi'an Jiaotong University

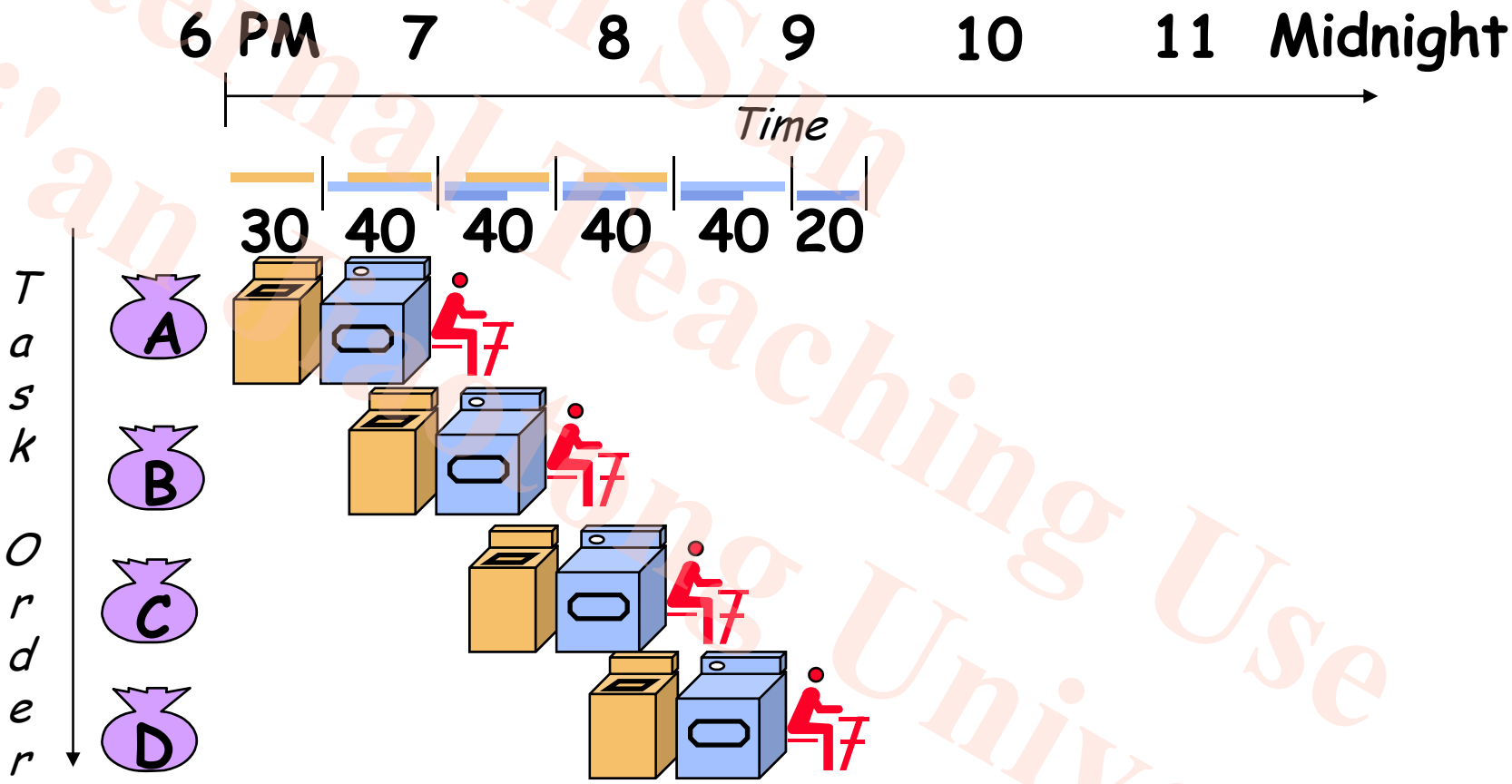
流水线的概念（2）



4人的衣量，如果顺序洗衣服，共需要6个小时。
如果她们学过流水线需要多少时间？

Hongbin Sun
International Teaching Use
Xi'an Jiaotong University

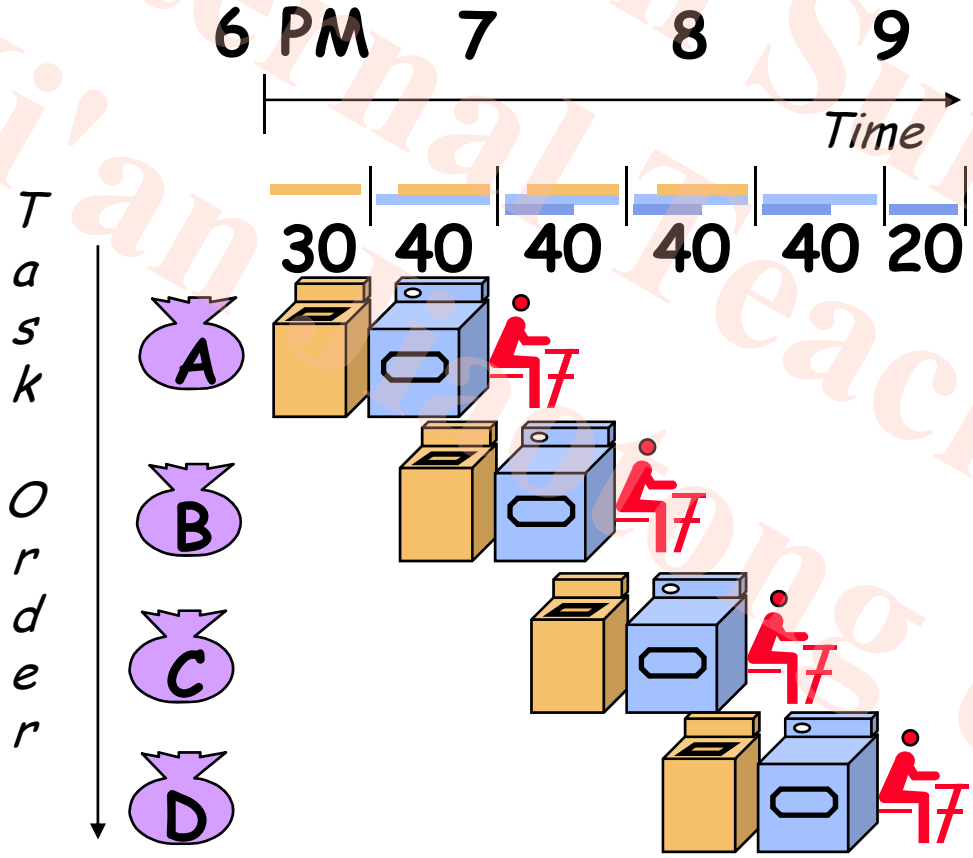
流水线的概念（3）



4人的衣量，如果用流水线的方式洗衣服，只需要3.5小时。

Hongbin Sun
Internal Teaching Use
Xi'an Jiaotong University

流水线的概念（4）

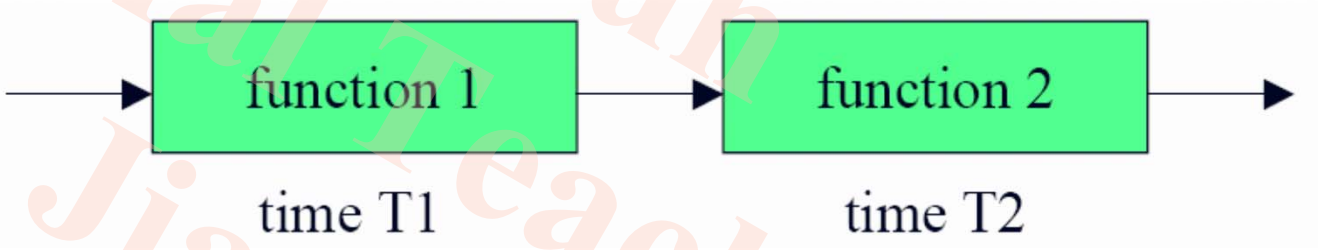


- 流水线无法减少每个任务的延迟，但可以提高整个工作的吞吐率。
- 流水线的速率受限于速度最慢的一级。
- 多个任务同时执行。
- 最大提速比=流水线级数。
- 进/出流水线的时间降低了提速比。

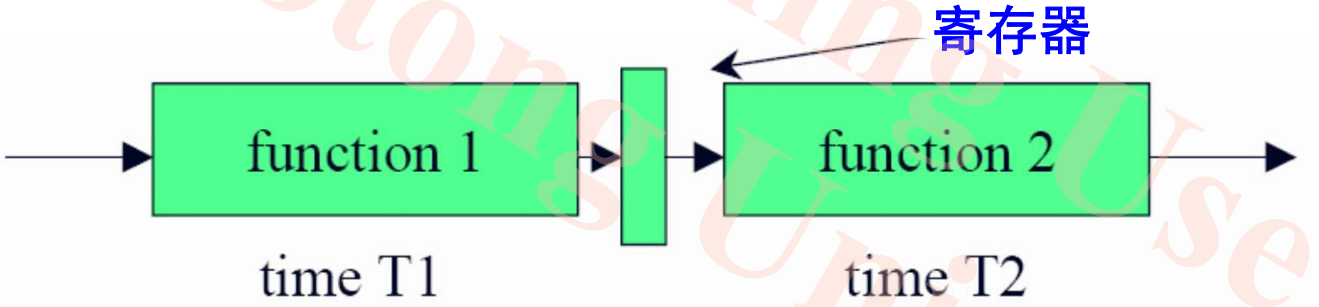
Hongbin Sun
Internal Teaching Use
Xi'an Jiaotong University

处理器的流水线

- 通常，最大的处理速度取决于计算逻辑的传输延迟。



- 如上图，每隔**T1+T2**时间周期可以处理一个输入数据



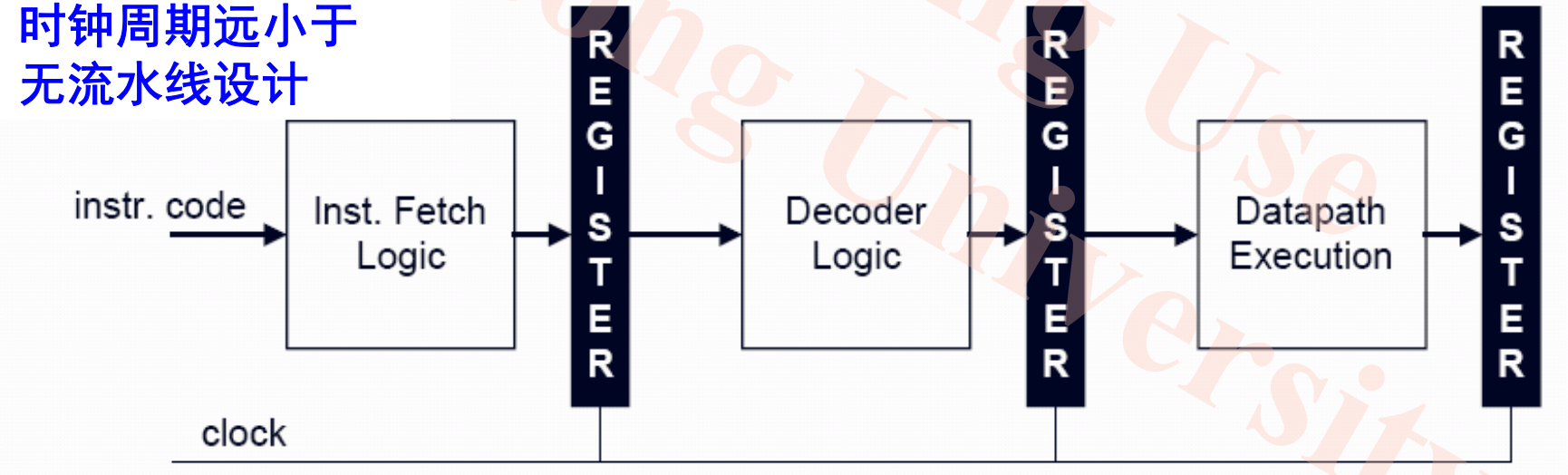
- 如上图的结构，每隔**max{T1,T2}**时间周期可以处理一个输入数据，但是每个输入处理时间仍需要**T1+T2**

Hongbin Sun
Internal Teaching Use
Xi'an Jiaotong University

ARM的流水线

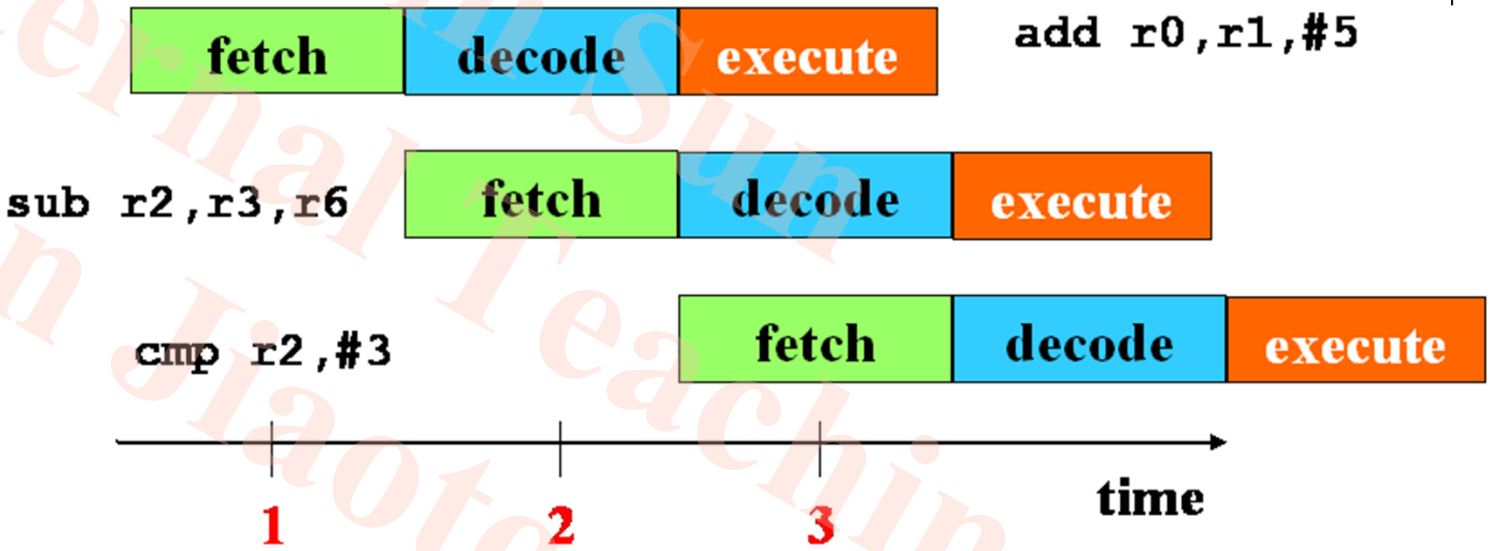
- ARM7使用三级流水线
  - 获取：从存储器中获取指令代码到指令流水线。
  - 译码：指令译码，通过控制信号操作数据通路准备执行。
  - 执行：指令获得数据通路“使用权”：读寄存器, 移位操作, ALU运算和写回。
- 每一级的结果存储在寄存器中。

时钟周期远小于
无流水线设计



Hongbin Sun
International Teaching Use
Xi'an Jiaotong University

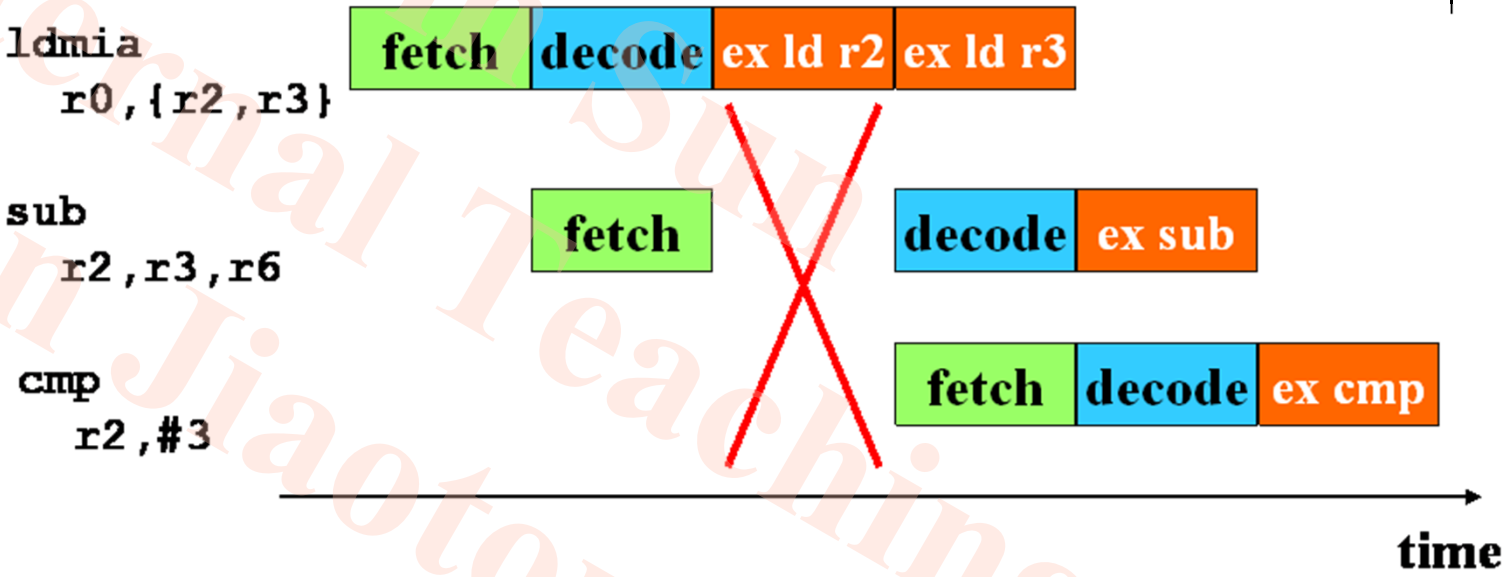
ARM的流水线（续）



- 任意时刻，有三条不同的指令占据着流水线的不同级
- 完成一条指令需要三个时钟周期，称之为三周期延迟
- 流水线装满后，处理器一个时钟周期可以完成一条指令。因此，流水线的吞吐率是每周期一条指令。
- 流水线提高了吞吐率，但无法降低延迟。

Hongbin Sun
Internal Teaching Use
Xi'an Jiaotong University

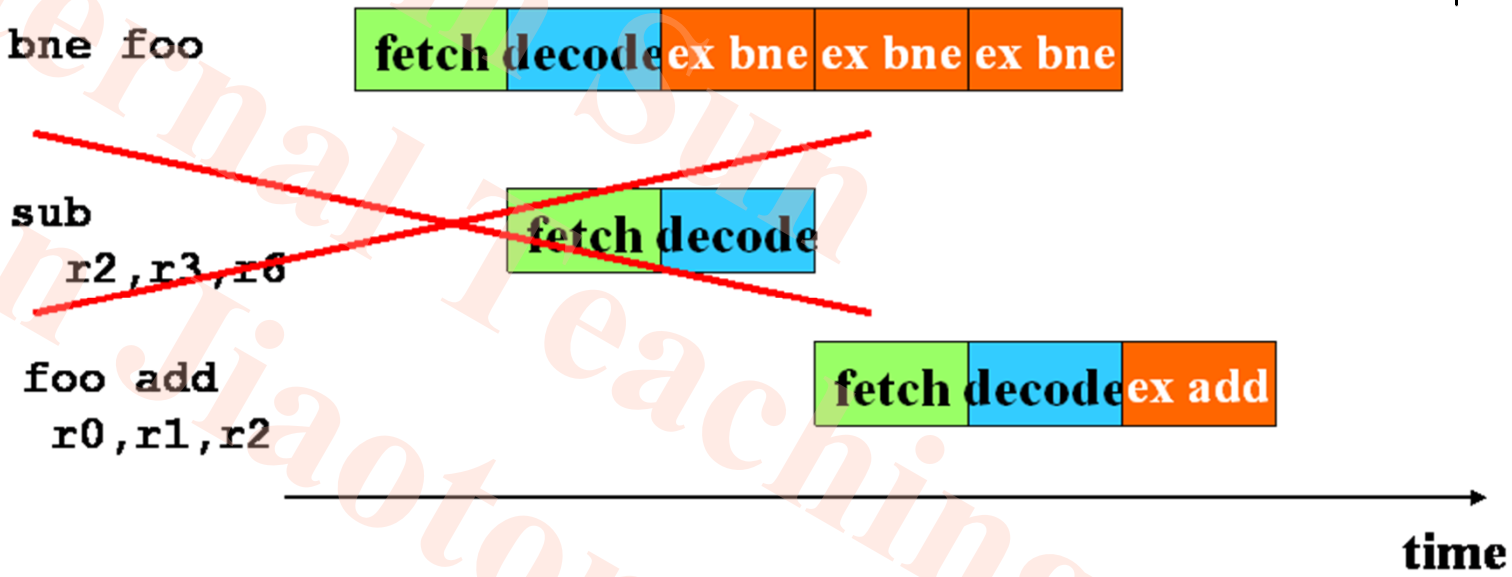
ARM的流水线数据阻滞



- 如果流水线的某一级无法在一个时钟周期内完成，将会导致**流水线阻滞**。
- 例如多寄存加载指令（**LDMIA**），将导致**数据阻滞（Data hazard）**。

Hongbin Shi
Internal Teaching Use
Xi'an Jiaotong University

ARM的流水线控制阻滞



- 分支指令将导致流水线产生**控制阻滞**，也被称为**分支损失（Branch Hazard）**。
- 是否执行分支指令**BNE**要在第三个时钟周期才能确定，因此，浪费两个时钟周期。

Hongbin Sun
Internal Teaching Use
Xi'an Jiaotong University

延迟分支（delayed branch）

- 为改善流水线效率，可使用延迟分支技术。
- 延迟分支要求分支指令后面的n条指令总是执行，无论分支指令执行与否。这样，在分支指令执行期间CPU能够让流水线满负载运行。
- 但是，分支指令后面可能是空操作（NOP）。因为处于延迟分支窗口中的指令对于两条执行路径都要有效。如没有没有足够的有效指令填充，则需使用NOP。

Hongbin Sun
Intel Mediaworks Use
Xi'an Jiaotong University

Four Branch Hazard Alternatives

#1: Stall until branch direction is clear

#2: Predict Branch Not Taken

- Execute successor instructions in sequence
- “Squash” instructions in pipeline if branch actually taken
- Advantage of late pipeline state update
- 47% MIPS branches not taken on average
- PC+4 already calculated, so use it to get next instruction

#3: Predict Branch Taken

- 53% MIPS branches taken on average
- But haven’t calculated branch target address in MIPS
  - MIPS still incurs 1 cycle branch penalty
  - Other machines: branch target known before outcome

Hongbin Sun
International Teaching Use
Xi'an Jiaotong University

Four Branch Hazard Alternatives

#4: Delayed Branch

- Define branch to take place AFTER a following instruction

```
branch instruction
  sequential successor1
  sequential successor2
  ........
  sequential successorn
branch target if taken
```

Branch delay of length *n*

- 1 slot delay allows proper decision and branch target address in 5 stage pipeline
- MIPS uses this

Hongbin Sun
Internal Teaching Use
Xi'an Jiaotong University

Delayed Branch

- Where to get instructions to fill branch delay slot?
  - Before branch instruction
  - From the target address: only valuable when branch taken
  - From fall through: only valuable when branch not taken
- Compiler effectiveness for single branch delay slot:
  - Fills about 60% of branch delay slots
  - About 80% of instructions executed in branch delay slots useful in computation
  - About 50% (60% x 80%) of slots usefully filled
- Delayed Branch downside: 7-8 stage pipelines, multiple instructions issued per clock (superscalar)

Hongbin Sun
Internal Teaching Use
Xi'an Jiaotong University

Scheduling Branch Delay Slots

A. From before branch

```
add  $1,$2,$3
if $2=0 then
delay slot
```

becomes

```
if $2=0 then
add $1,$2,$3
```

B. From branch target

```
sub $4,$5,$6

add  $1,$2,$3
if $1=0 then
delay slot
```

becomes

```
add  $1,$2,$3
if $1=0 then
sub $4,$5,$6
```

C. From fall through

```
add  $1,$2,$3
if $1=0 then
delay slot

sub $4,$5,$6
```

becomes

```
add  $1,$2,$3
if $1=0 then
sub $4,$5,$6
```

- A is the best choice, fills delay slot & reduces instruction count (IC)
- In B, the sub instruction may need to be copied, increasing IC
- In B and C, must be okay to execute sub when branch fails

Hongbin Sun
Internal Teaching Use
Xi'an Jiaotong University

范例：ARM中for循环的执行时间

- 决定FIR滤波器应用的执行时间：
  - for (I = 0; i < N; i++)
  - f = f + c[i] * x[i];

Hongbin Sun
Internal Teaching Use
Xi'an Jiaotong University

FIR滤波器汇编程序

```
; loop initiation code
MOV r0, #0        ; use r0 for i, set to 0
MOV r8, #0        ; use a separate index for arrays
ADR  r2, N        ; get address for N
LDR  r1, [r2]     ; get value of N for loop termination test
MOV r2, #0        ; use r2 for f, set to 0
ADR  r3, c        ; load r3 with address of base of c array
ADR  r5, x        ; load r5 with address of base of x array
          ; loop body
loop      LDR r4, [r3, r8]        ; get value of c[i]
          LDR r6, [r5, r8]        ; get value of x[i]
          MUL r4, r4, r6          ; compute c[i]*x[i]
          ADD r2, r2, r4          ; add into running sum
          ; update loop counter and array index
          ADD r8, r8, #4          ; add one to array index
          ADD r0, r0, #1          ; add 1 to i
          ; test for exit
          CMP r0, r1
          BLT loop                ; if i < N, continue loop
loopend ...
```

Hongyan Yan
Internal Teaching Use
Xi'an Jiaotong University

FIR滤波器的性能

Block	Variable	# instructions	# cycles
Initialization	t_{init}	7	7
Body	t_{body}	4	4
Update	t_{update}	2	2
Test	t_{test}	2	[2,4]

$$t_{loop} = t_{init} + N(t_{body} + t_{update}) + (N-1)\, t_{test,worst} + t_{test,best}$$

继续循环是最坏情况

退出循环是最好情况

判断一定跳转？？？

Hongbin Sun
Internal Teaching Use
Xi'an Jiaotong University

思考

- 思考前面课程提到的问题：
  - “条件执行指令只有在条件序列小于等于3的时候才会达到高效，当大于时，要使用条件转移指令。” ---为什么???

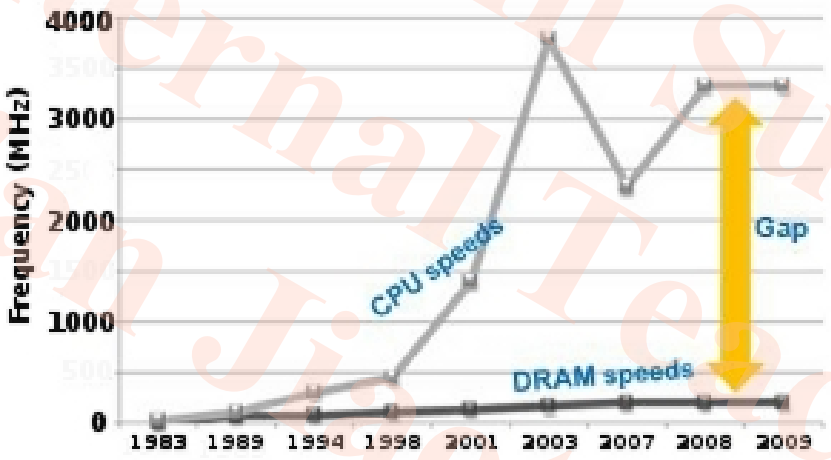
Hongbin Shu
Internal Teaching Use
Xi'an Jiaotong University

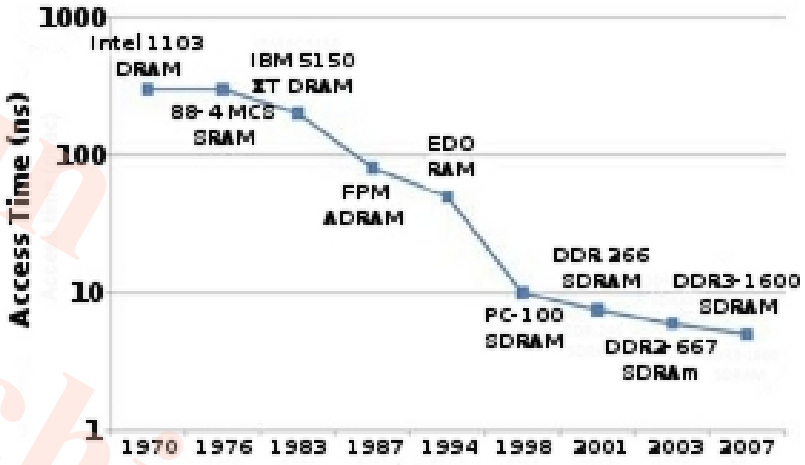
2 高速缓存

- 静态随机存储器（**SRAM**）：依靠一对反相器存储数值，速度很快，但面积占用比**DRAM**高。
- 动态随机存储器（**DRAM**）：依靠电容电荷实现数据存储（必须刷新），很小但比**SRAM**速度慢（**5~10**倍）
- 存储器性能 **vs** 逻辑性能**:**
  - 存储器比逻辑电路的性能提升慢得多（“存储墙”问题）。
  - 因此，速度非常快的存储器是很昂贵的。
  - 同时，应用对存储器的需求量越来越大。
- 用户需要大容量且快速的存储资源。
  - **SRAM**访问延迟**:2~25ns**，耗费**:$100~$250 per MB**
  - **DRAM**访问延迟**:60~120ns**，耗费**:$0.05~$0.1 per MB**
  - **Disk**访问延迟**:10~20million ns**，耗费**:$0.05~$0.1 per MB**

Hongbin Cam
Internal Teaching Use
Xi'an Jiaotong University

“存储墙”问题



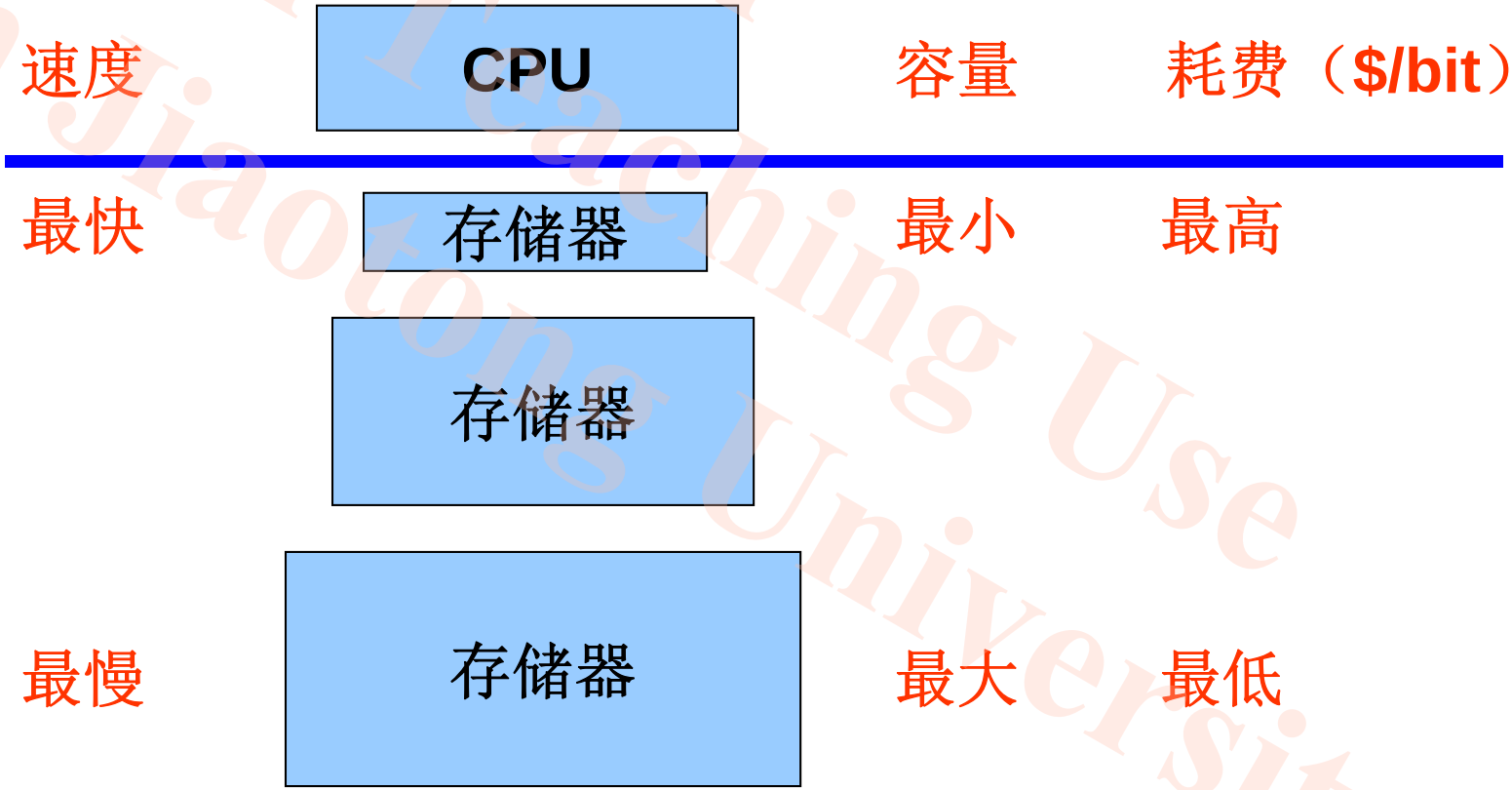


- 随着集成电路工艺尺寸的缩小，处理器的速度按照“摩尔定律”每隔18个月就翻一番。
- 存储器的速度每年仅增长7%。
- 处理器与存储器之间的性能鸿沟每21个月就翻一番。

Hongbin Sun Internal Teaching Use Xi'an Jiaotong University

存储系统（Memory Hierarchy）

- 如何组织存储器资源，但又不显著提高耗费？
- 答案是：构建存储系统。



Hongbin Sun
Internal Teaching Use
Xi'an Jiaotong University

存储系统

靠近处理器

存储类型	容量	访问速度
Processor Registers	64 to 256 bytes	1 - 5 nsec
On-chip cache	8 - 32 Kbytes	~ 10 nsec
Second-level cache	128 - 512 Kbytes	10's nsec
Main memory (DRAM)	16M - 4G bytes	~ 100 nsec
Disk or other store	10's - 100's Gbytes	10's - 100's msec

高速缓存保存主存中最近用到的内容的拷贝。

Hongbin Sun
Internal Teaching Use
Xi'an Jiaotong University

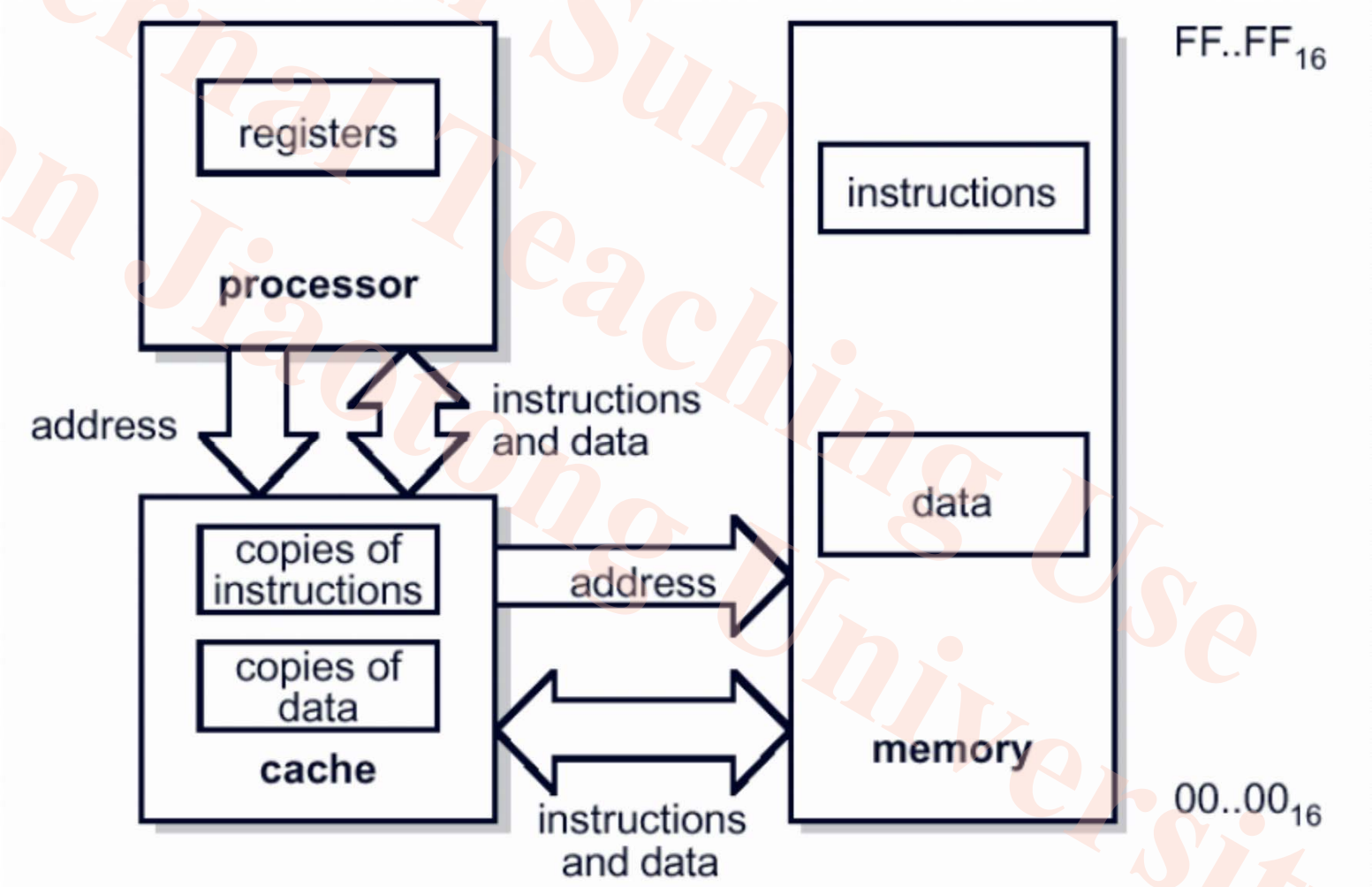
局部性原理

- 局部性原理是存储系统层级构成的基础。
  - 时域局部性：当前被访问的数据有可能很快再次被访问；
  - 空域局部性：当前被访问数据地址相邻的数据有可能很快被访问。
- 程序为何会具有局部性？
- 高速缓存减少了内存的平均访问时间，但也提高了内存访问时间的可变性，值得注意！

Hongbin Sun
Internal Teaching Use
Xi'an Jiaotong University

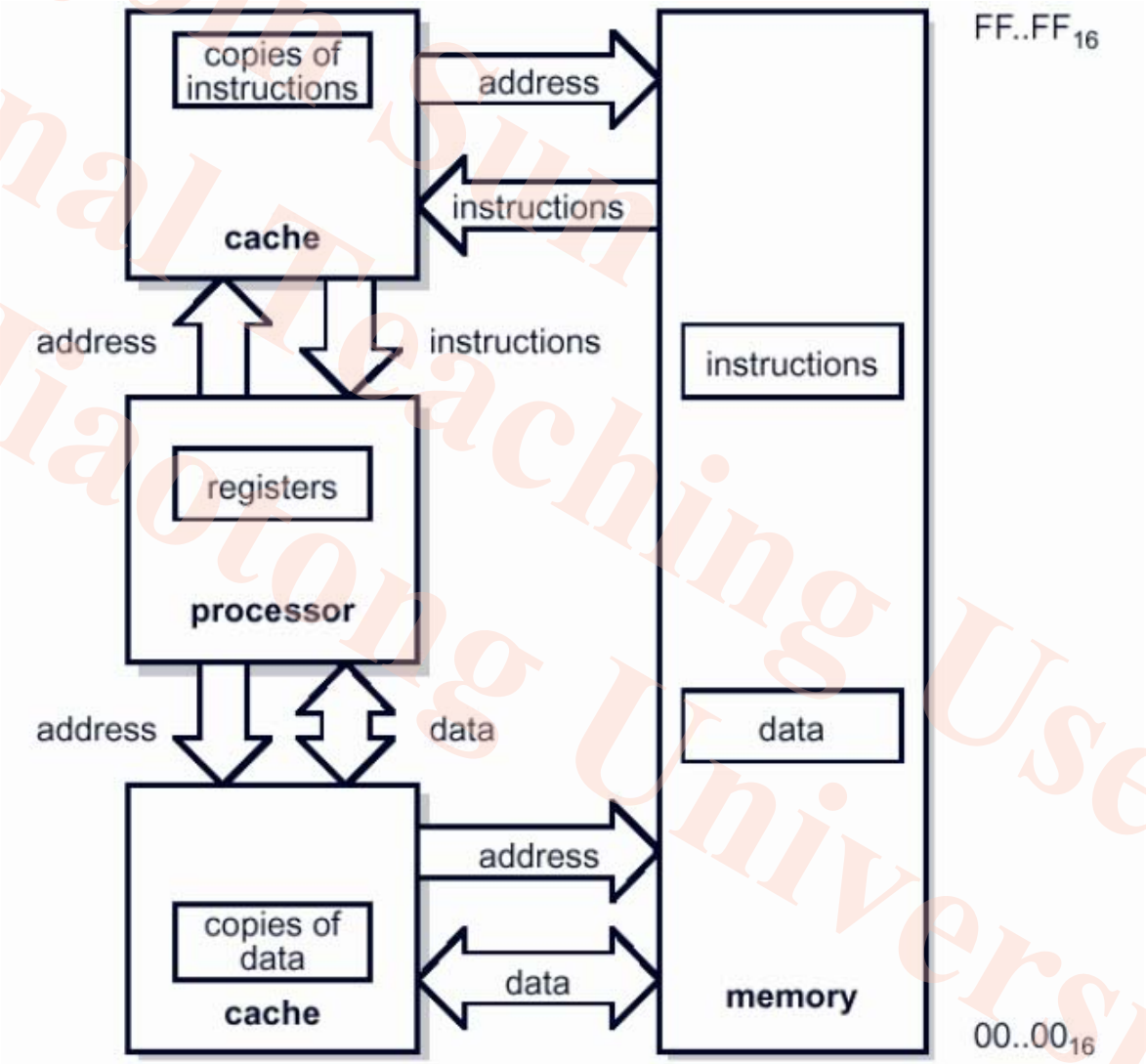
统一的高速缓存

数据与指令之间共享同一个高速缓存。



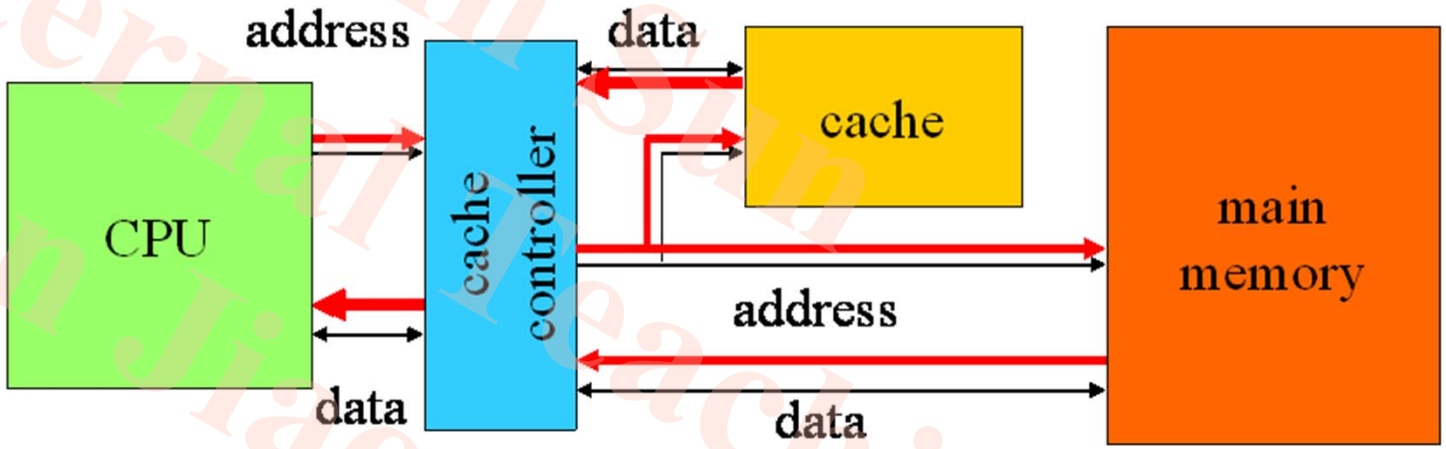
Hongbin Sun
Internal Teaching Use
Xi'an Jiaotong University

哈佛高速缓存



Hongqing Sun
Internal Teaching Use
Xi'an Jiaotong University

Cache的基本术语



- 数据块（**data block**）：数据传送的最小单元 – 也被称为**cache line**；
- 高速缓存命中（**cache hit**）：被请求的数据在高速缓存中；
- 高速缓存未命中（**cache miss**）：被请求的数据不在高速缓存中。
- 工作集（**working set**）：**CPU**在一段时间内访问的活动单元集合。

Hongbin Sun
Internal Teaching Use
Xi'an Jiaotong University

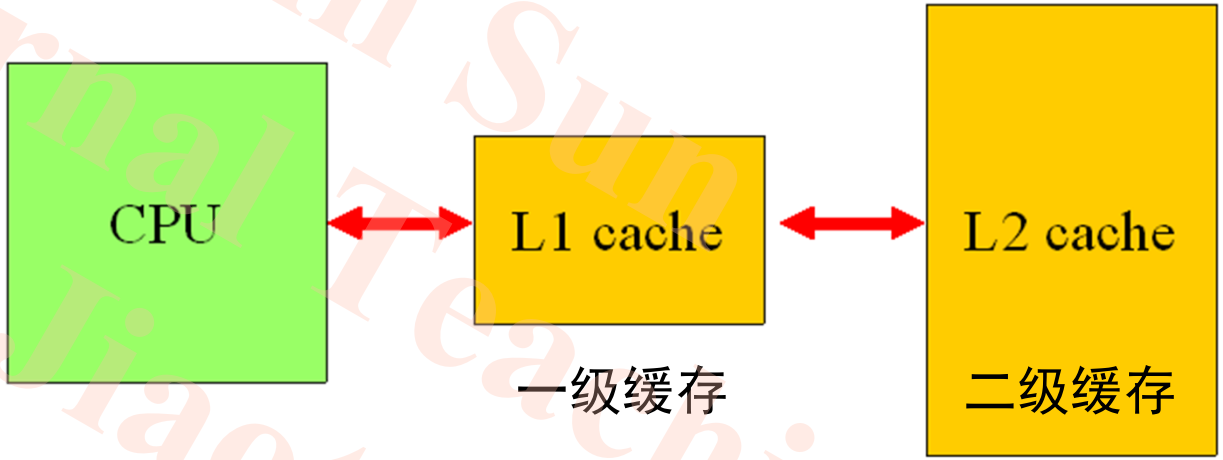
未命中的类型划分

- **强制性未命中**（**compulsory miss**），也称为**冷未命中**（**cold miss**），发生在单元第一次被访问时。
- **容量未命中**（**capacity miss**）的发生是由于工作集过大。
- **冲突未命中**（**conflict miss**）的发生是由于两个地址映射到高速缓存的同一个单元。

- 平均内存访问时间计算公式：
- $t_{av} = ht_{cache} + (1-h)t_{main}$, h代表命中率

Hongbin Sun
Internal Teaching Use
Xi'an Jiaotong University

多级高速缓存



- 平均内存访问时间计算公式：
- $$t_{av} = h_1 t_{L1} + h_2 t_{L2} + (1 - h_2 - h_1) t_{main}$$
- h_1表示一级缓存命中率，h_2表示命中二级缓存而未命中一级缓存的概率。

Hongbin Sun
Internal Teaching Use
Xi'an Jiaotong University

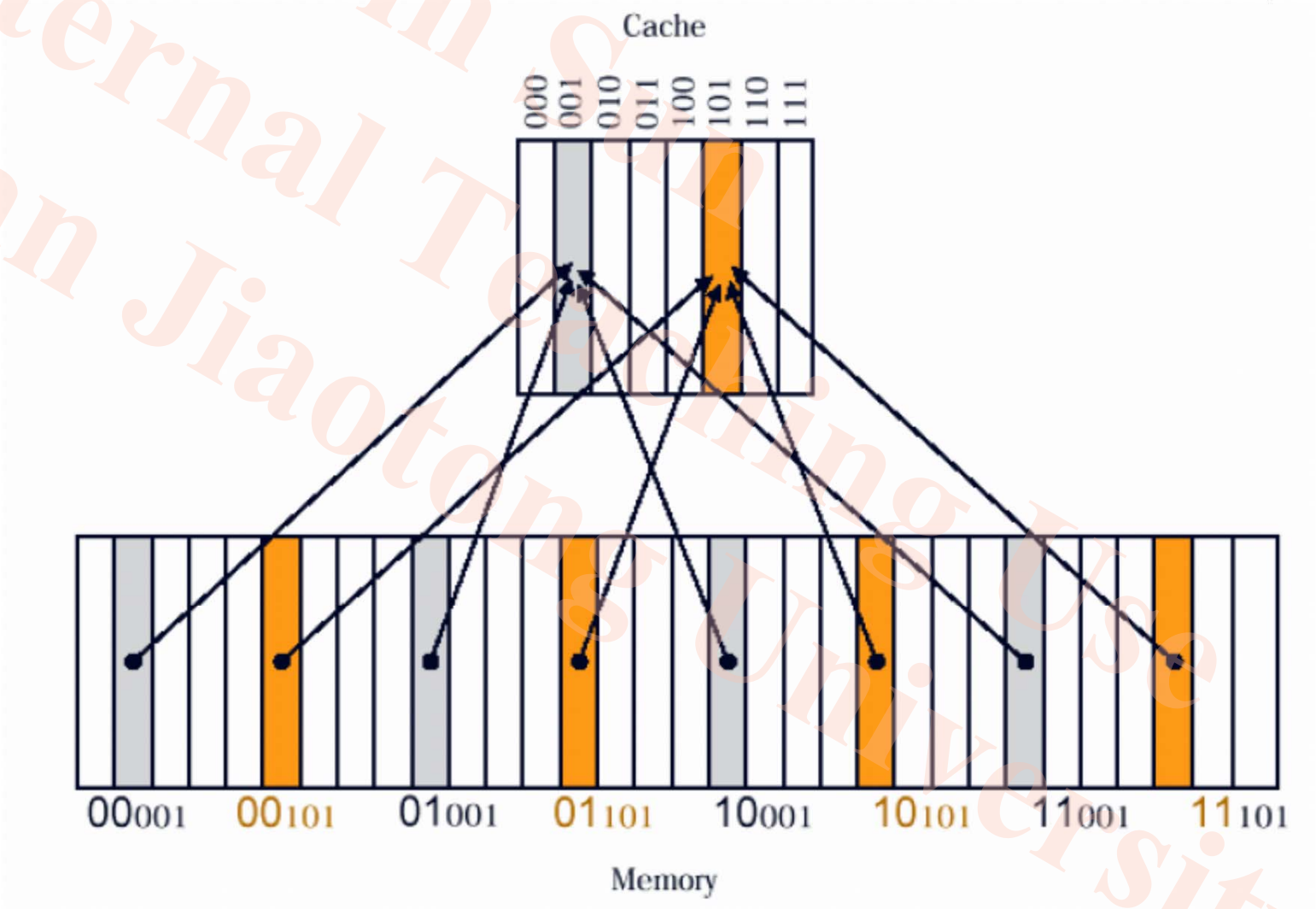
高速缓存设计

- 让我们思考两个问题：
  - 我们如何知道需要的数据是否在高速缓存中？（命中还是未命中？）
  - 如果在高速缓存中，我们如何找到它？
- 我们先看一个最简单的例子：
  - 数据块的大小为一个字节；
  - **直接映射（direct mapped）**高速缓存

直接映射指低一级存储中的每一个数据在高速缓存中的映射都是唯一确定的地址。

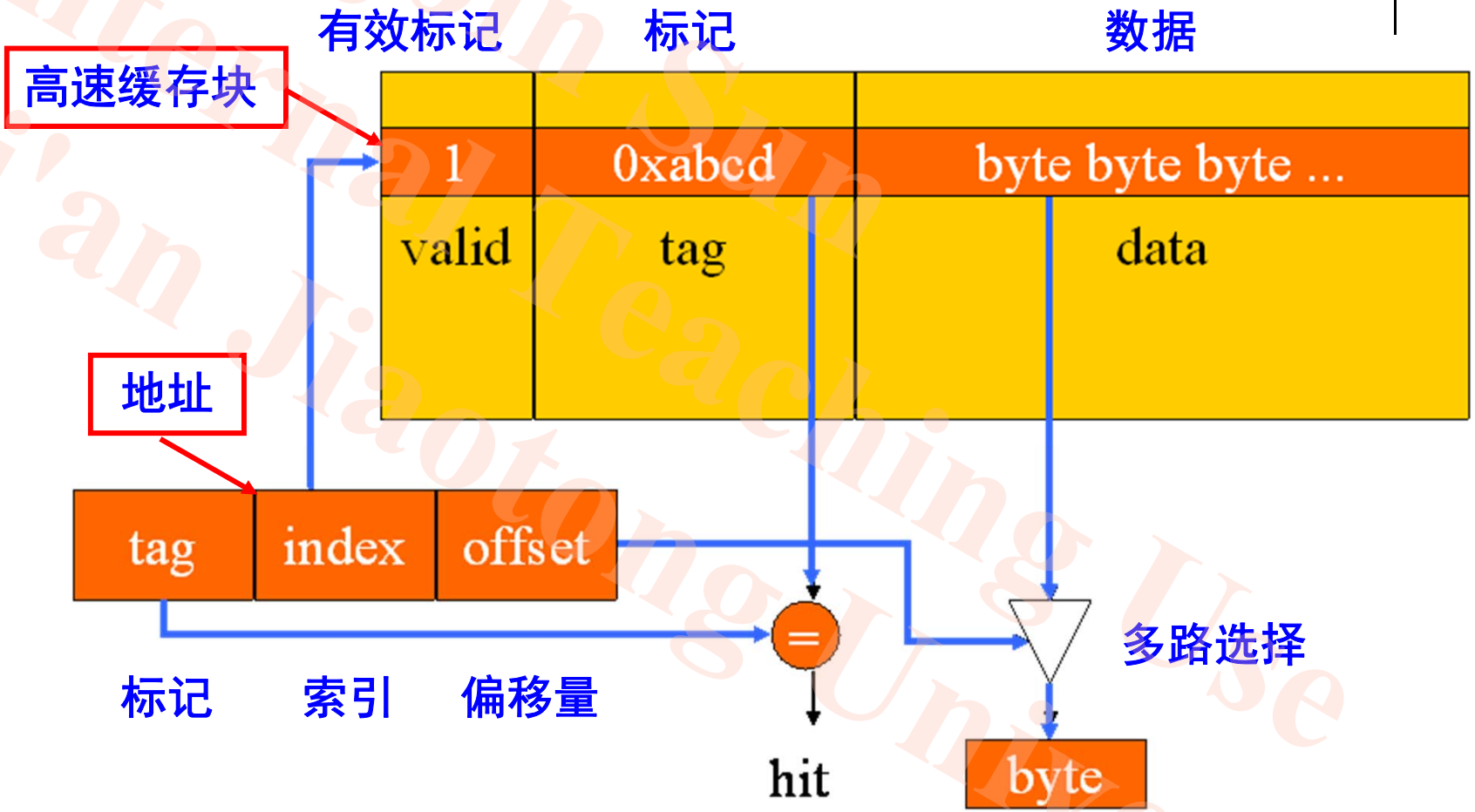
Hongbin Sun
Internal Teaching Use
Xi'an Jiaotong University

直接映射高速缓存图示



Hongbin Sun
Internal Teaching Use
Xi'an Jiaotong University

直接映射高速缓存结构



只有一个块需要检查，索引唯一地确定了要检查的块。

Hongqin Shi
Internal Teaching Use
Xi'an Jiaotong University

直接映射高速缓存示例（1）

address	data
000	0101
001	1111
010	0000
011	0110
100	1000
101	0001
110	1010
111	0100

Hongbin Sun
Internal Teaching Use
Xi'an Jiaotong University

直接映射高速缓存示例（2）

- After 001 access:

block	tag	data
00	-	-
01	0	1111
10	-	-
11	-	-

- After 010 access:

block	tag	data
00	-	-
01	0	1111
10	0	0000
11	-	-

Hongbin Sun
Internal Teaching Use
Xi'an Jiaotong University

直接映射高速缓存示例（3）

- After 011 access:

block	tag	data
00	-	-
01	0	1111
10	0	0000
11	0	0110

- After 100 access:

block	tag	data
00	1	1000
01	0	1111
10	0	0000
11	0	0110

Hongbin Sun
Internal Teaching Use
Xi'an Jiaotong University

直接映射高速缓存示例（4）

- After 101 access:

block	tag	data
00	1	1000
01	1	0001
10	0	0000
11	0	0110

- After 111 access:

block	tag	data
00	1	1000
01	1	0001
10	0	0000
11	1	0100

Hongbin Sun
Internal Teaching Use
Xi'an Jiaotong University

写操作

- 写操作比读操作复杂，因为我们要更新高速缓存和主存（或下一级存储）的内容。
- 两种写操作方式：
  - 通写（**write through**）：每次写操作都将同时改变高速缓存和主存单元。这种模式保证了高速缓存和主存的一致性，但会产生额外的主存通信。
  - 回写（**write back**）：只是在将某一单元从高速缓存中移出时才进行写操作（利用**dirty bit**来判断）。这种模式可以减少那些该单元移出高速缓存之前对它进行的多次写操作。

Hongbin Sun
Intelligent Teaching Lab
Xi'an Jiaotong University

直接映射高速缓存操作简述

- A particular memory item is stored in a unique location in the cache.
- To check if a particular memory item is in cache, the relevant address bits are used to access the cache entry.
- The top address bits are then compared with the stored tag. If they are equal, we have got a hit.
- Two items with the same cache address field will contend for use of that location.
- Only those bits of the address which are not used to select within the line or to address the cache RAM need be stored in the tag field.
- When a miss occurs, data cannot be read from the cache. A slower read from the next level of memory must take place, incurring a miss penalty.
- A cache line is typically more than one word. It shows 4 words in the diagram here. A large cache line exploits principle of spatial locality - more hits for sequential access. It also incurs higher miss penalty.

Hongbin Sun
Internal Teaching Use
Xi'an Jiaotong University

命中 vs. 未命中

- 读命中：
  - 所希望的情况：高速缓存直接传送数据到处理器。
- 读未命中：
  - 暂停CPU，从主存中获取数据块，传送到高速缓存，重新启动
- 写命中：
  - 可以替换高速缓存和主存中的数据（通写）
  - 只将数据写入高速缓存（稍后写入主存）
- 写未命中：
  - 将整个数据块读入高速缓存，然后写某个字。

Hongbin Sun
Internal Teaching Use
Xi'an Jiaotong University

直接映射的局限性

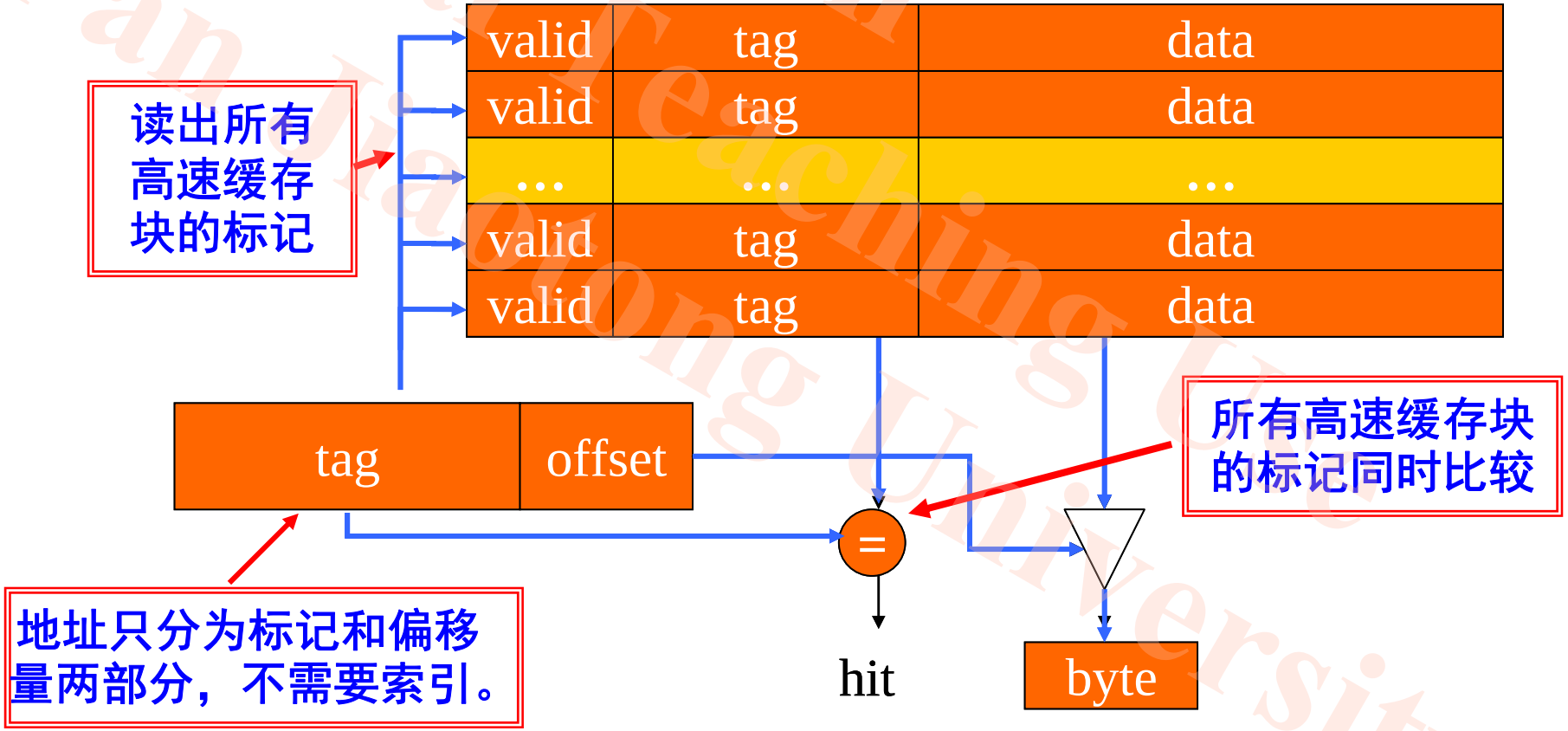
- 直接映射高速缓存速度快，耗费相对较低。
- 直接映射高速缓存映射策略简单，有一定的局限性。
- 容易造成冲突未命中的情况：
  - **Array a[] uses locations 0, 1, 2, …**
  - **Array b[] uses locations 1024, 1025, 1026, …**
  - 计算 **a[i] + b[i]** 将产生冲突未命中的情况.
- 如果频繁访问的单元正好被映射到同一个高速缓存块，我们将无法充分利用高速缓存的优势。

如何解决？？？

Hongbin Sun
Internal Teaching Use
Xi'an Jiaotong University

全相联高速缓存

- 全相联（**fully-associative**）高速缓存：任意个内存数据块可以映射到高速缓存中的任意个地址。



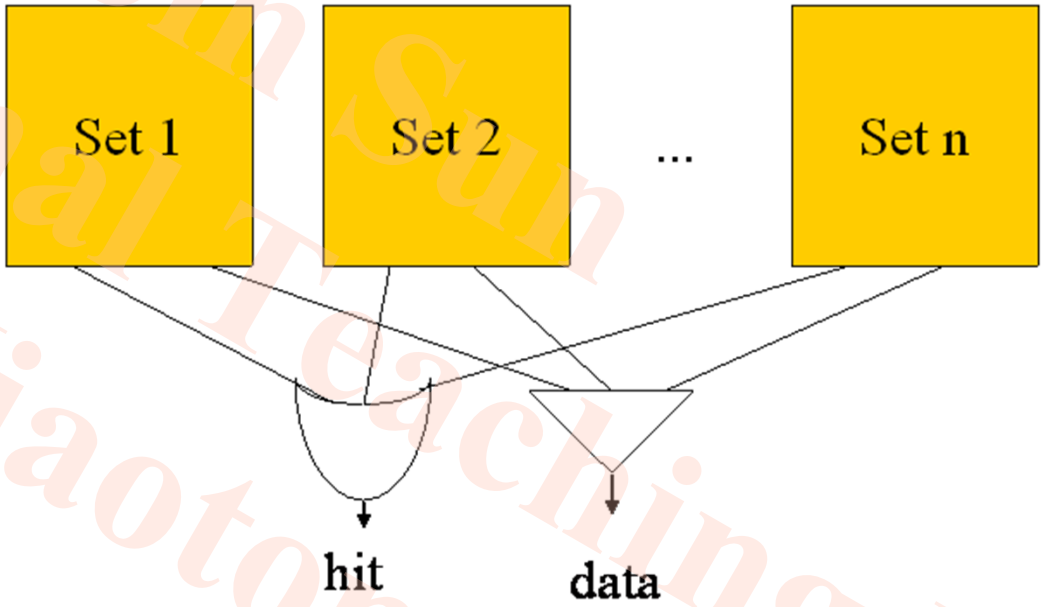
Hongbin Sun International Teaching Base Xi'an Jiaotong University

全相联的局限性

- 全相联不限制主存数据在高速缓存中的存放位置，使映射具有最大的灵活性，命中率最高。
- 但是，由于全相联通常需要采用内容可寻址存储器（CAM），耗费很高。同时，要在每次数据访问时读出并比较所有标记单元，速度慢，能耗也非常大。
- 因此，全相联高速缓存一般只用于设计小容量缓存器。

Hongbin Sun
Intelligent Teaching Use
Xi'an Jiaotong University

组相联高速缓存



- 组相联（**Set Associative**）是直接映射和全相联之间的折中。组（**Set**）由共享同一索引的所有块组成。
- 由每个组的高速缓存块个数来标识，**N**路组相联。
- 高速缓存访问请求按照索引广播给组内所有缓存块，判断是否命中。每个组由一个全相联高速缓存实现。

Hongbin Sun
Internal Teaching Use
Xi'an Jiaotong University

替换策略

- 替换策略：在高速缓存“满”且需要载入新的数据块时，需要考虑何种旧数据块需要被移出高速缓存。
- 直接映射不存在替换策略问题，因为每个数据块对应唯一映射地址，其替换的数据块也是确定的。
- 组相联与全相联高速缓存需要考虑替换策略：
  - 随机替换（**Random**）：在组内或全相联高速缓存中随机选择一个数据块被替换。优点：硬件简单，耗费低。缺点：命中率相对较低。
  - 最近最少使用策略（**Least Recently Used, LRU**）：将组内或全相联高速缓存中最近最少使用的数据块。优点：命中率高。缺点：硬件耗费大。 **Trade off**
  - 先入先出策略（**First In First Out, FIFO**）：折中！

Hongbin Cui
Interdisciplinary Leading Use
Xi'an Jiaotong University

两路组相联高速缓存示例（1）

address	data
000	0101
001	1111
010	0000
011	0110
100	1000
101	0001
110	1010
111	0100

Hongbin Sun
Internal Teaching Use
Xi'an Jiaotong University

两路组相联高速缓存示例（2）

- Final state of cache (twice as big as direct-mapped):

set	blk 0 tag	blk 0 data	blk 1 tag	blk 1 data
00	1	1000	-	-
01	0	1111	1	0001
10	0	0000	-	-
11	0	0110	1	0100

Hongbin Sun
Internal Teaching Use
Xi'an Jiaotong University

两路组相联高速缓存示例（3）

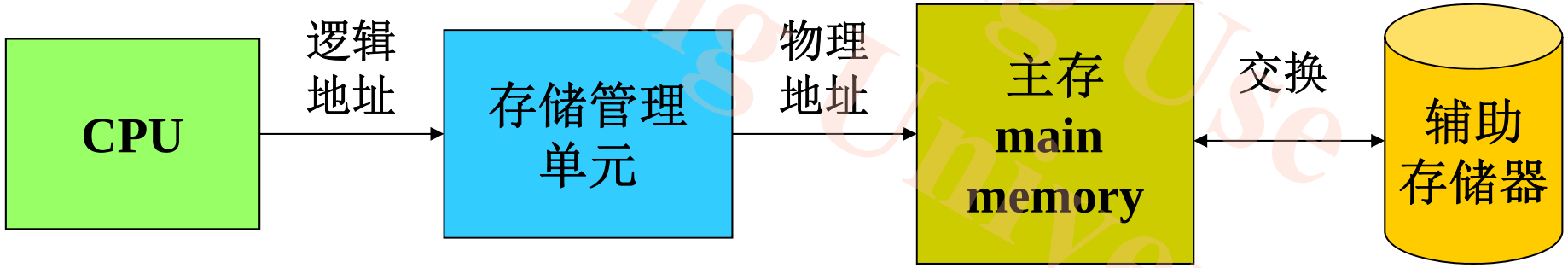
- Final state of cache (same size as direct-mapped):

set	blk 0 tag	blk 0 data	blk 1 tag	blk 1 data
0	01	0000	10	1000
1	10	0111	11	0100

Hongbin Sun
Internal Teaching Use
Xi'an Jiaotong University

3 存储管理单元和地址转换

- 存储管理单元在CPU和物理主存之间，实现逻辑地址到物理地址之间的转换。转换过程也称内存映射。
  - 物理地址（**physical address**）：用于内存芯片级的单元寻址，与实际的RAM单元对应。
  - 逻辑地址（**logical address**）：程序的抽象地址空间而不与实际的RAM单元对应。



（Memory Management Unit, MMU）

Hongbin Sun
Intel Lead Teaching Use
Xi'an Jiaotong University

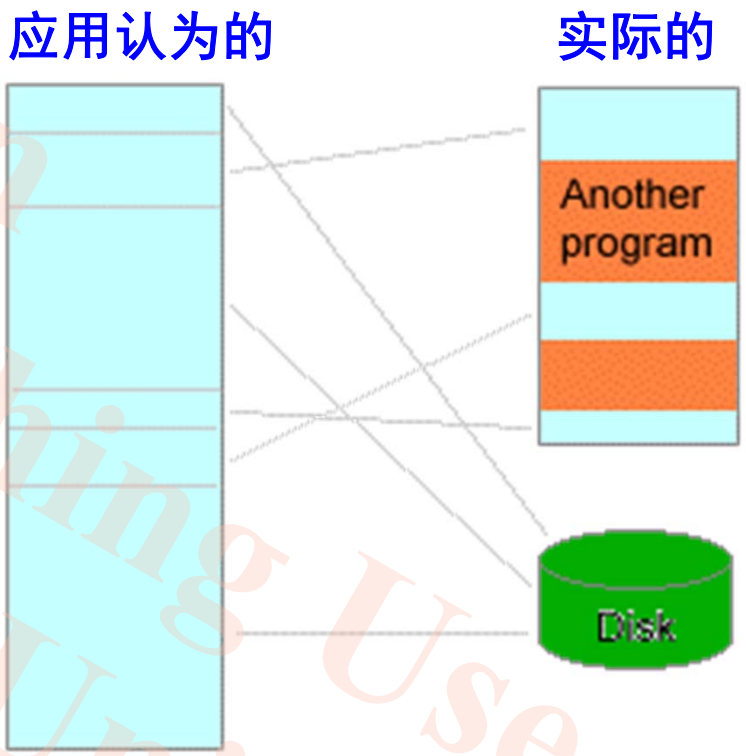
虚拟内存

- 什么是虚拟内存？？？
- 虚拟内存能做什么？？？

Hongbin Sun
Internal Teaching Use
Xi'an Jiaotong University

虚拟内存与虚拟寻址

- 虚拟内存是计算机系统内存管理的一种技术。它使得应用程序认为它拥有连续的可用的内存（一个连续完整的地址空间），而实际上，它通常是被分隔成多个物理内存碎片，还有部分暂时存储在外部磁盘存储器上，在需要时进行数据交换。
  -- from wikipedia



虚拟寻址（**virtual addressing**）**:**将逻辑地址转换成物理地址的寻址模式。

Hongbin Gan
Internal Teaching Use
Xi'an Jiaotong University

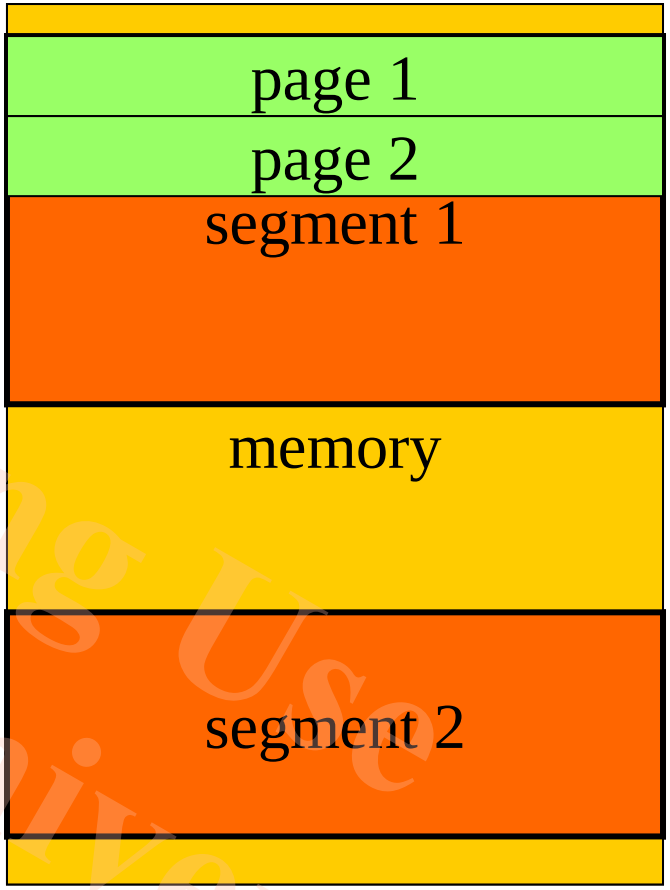
存储管理单元

- 存储管理单元实现的功能：
  - 实现虚拟寻址：用表把逻辑地址转换成物理地址。
  - 可以将部分程序移入辅助存储器，当程序执行需要时，再将镜像调入主存。
  - 缺页（page fault）异常：CPU请求一个不在主存的地址。
- 缺页异常重新启动的条件：
  - 所有内容已读到主存中；
  - MMU表已经进行了相应的更新。
- LRU替换策略。

Hongbin Sun
Internal Teaching Use
Xi'an Jiaotong University

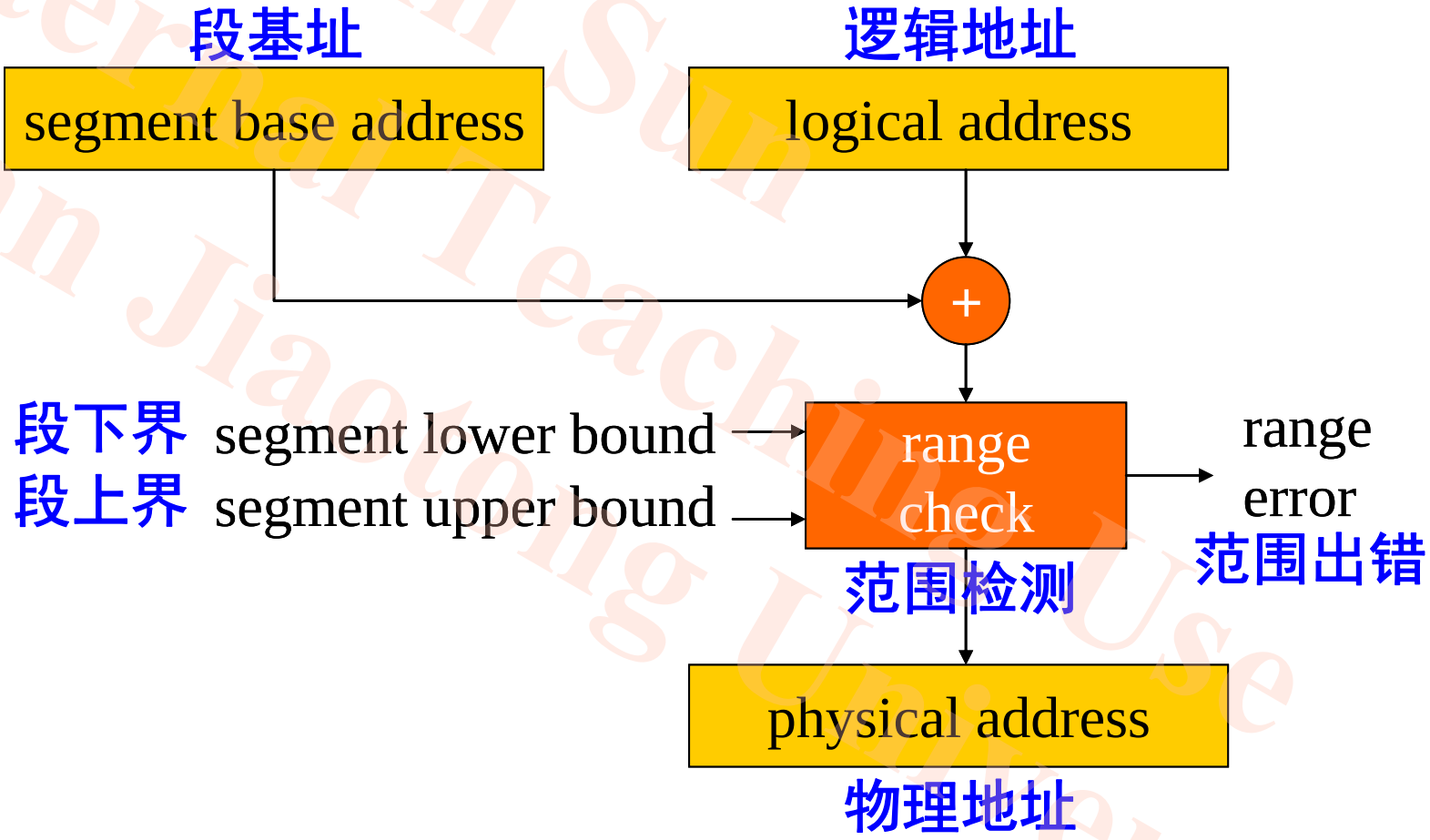
地址转换方法

- 需要某种寄存器或表来实现逻辑地址到物理地址之间的任意转换。
- 两种基本地址转换方法：
  - 分段：支持较大的、大小可不等的内存区域。
    - 起始地址+大小。
  - 分页：支持较小的、大小相等的内存区域。
- 段页式寻址模式：将每个段分成页并且其地址转换分成两步的方法来建立。



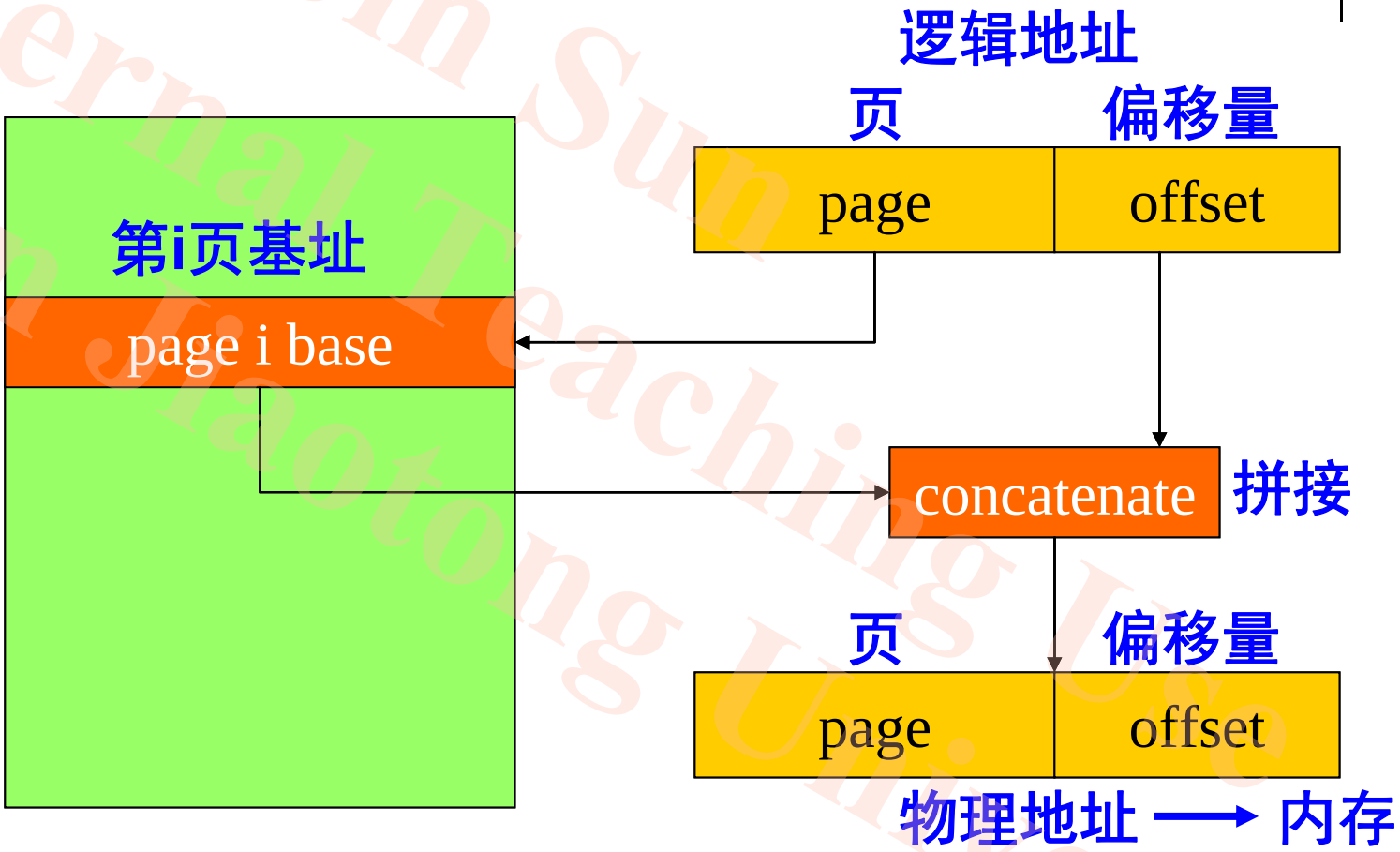
Hongbin Sun
Intelligent Teaching Used
Xi'an Jiaotong University

段地址转换



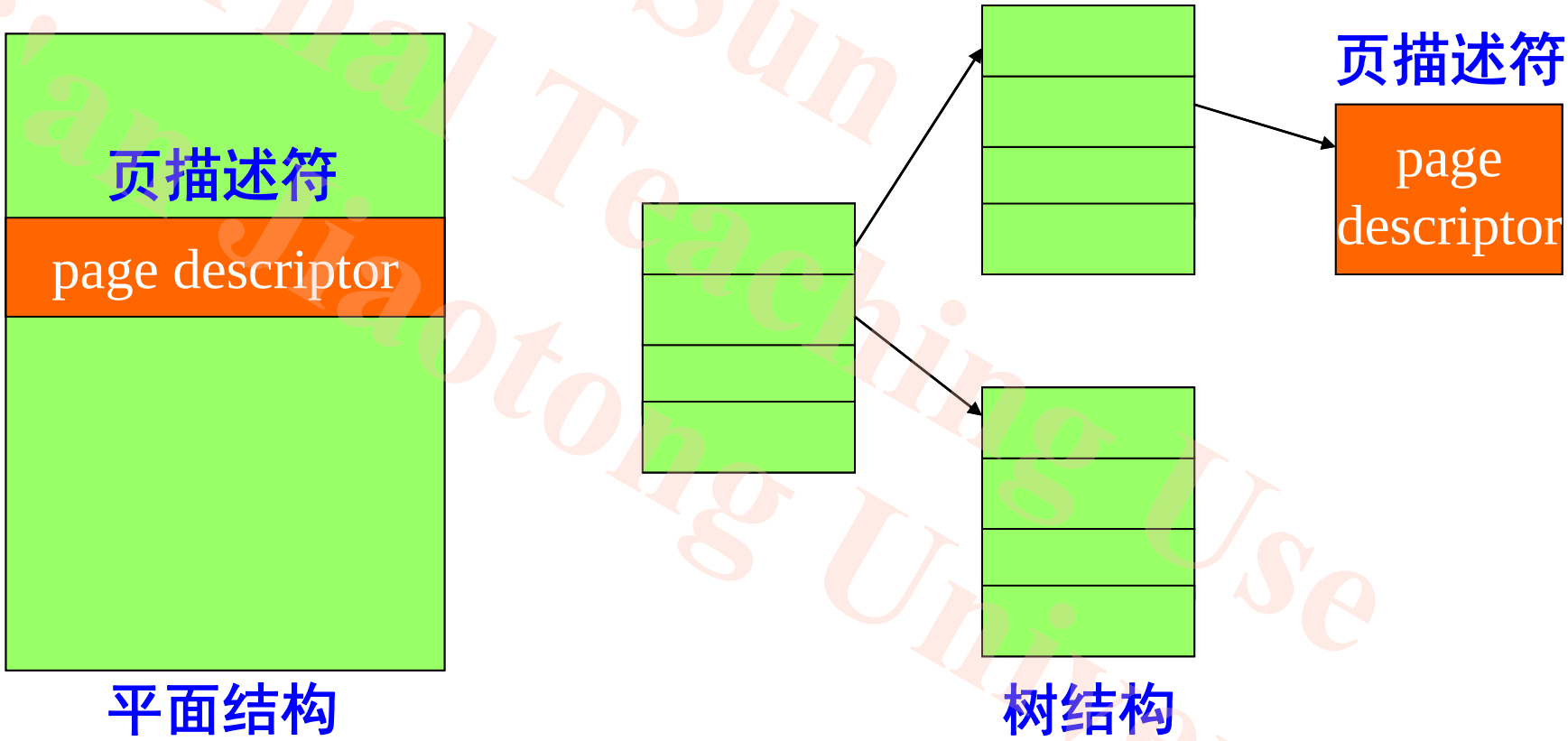
Hongbin Sun
Internal Teaching Use
Xi'an Jiaotong University

页地址转换



Hongbin Sun Internal Teaching Use Xi'an Jiaotong University

页表组织模式



Hongbin Sun
Internal Teaching Use
Xi'an Jiaotong University

转换后援缓冲区

- 大的转换表需要访问主存。
- 转换后援缓冲区（**Translation Lookaside Buffer, TLB**）：缓存转换表，通常很小。

Virtual Address	Physical Address	Dirty	Ref	Valid	Access

- **Valid**：表示逻辑段或逻辑页当前是否在物理内存。
- **Dirty**：表示该段或该页是否已被改写。
- **Ref&Access**：权限位。

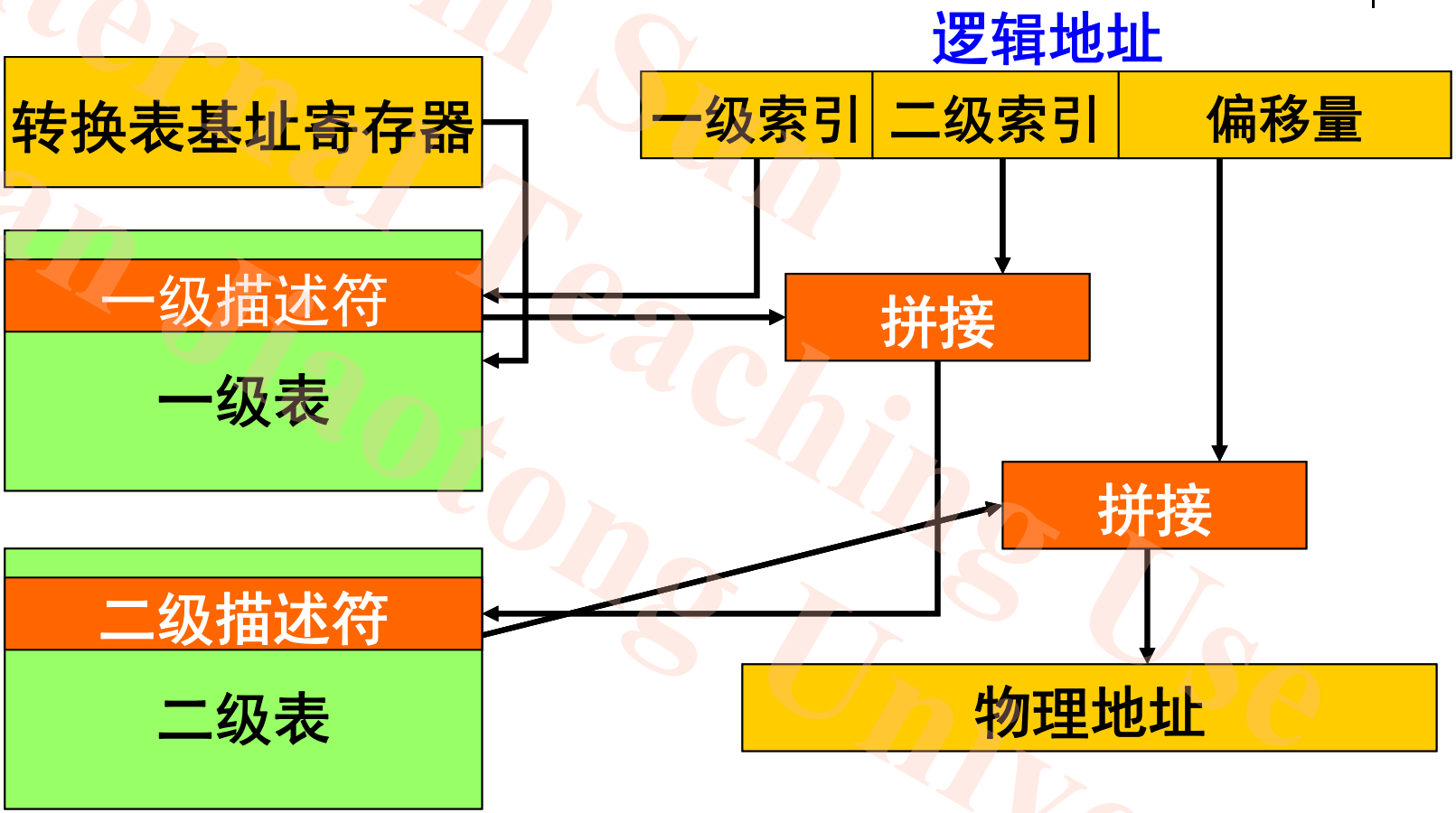
Hongbin Sun
Intel Teaching Lab
Xi'an Jiaotong University

ARM的内存管理

- 存储管理单元是一个可选部件，支持虚拟地址转换和内存保护。ARM的MMU支持下列内存区域类型：
  - 1MB内存块大小的段
  - 64KB的大页
  - 4KB的小页
- 地址可以在选择段映射或者页映射。
- 采用二级地址转换模式。

Hongbin Sun
Internal Teaching Use
Xi'an Jiaotong University

ARM两级地址转换



Hongbin Sun
Internal Teaching Use
Xi'an Jiaotong University