计系3 期末速通教程

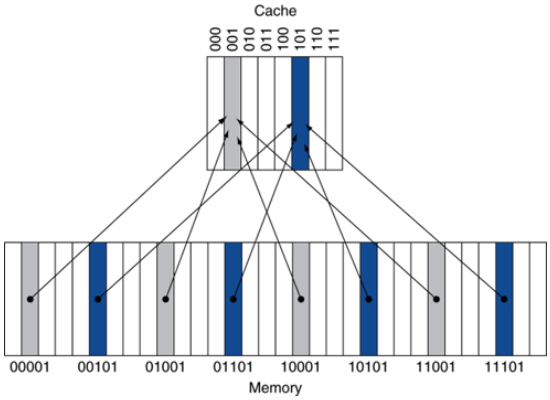
5. 存储

5.1 cache

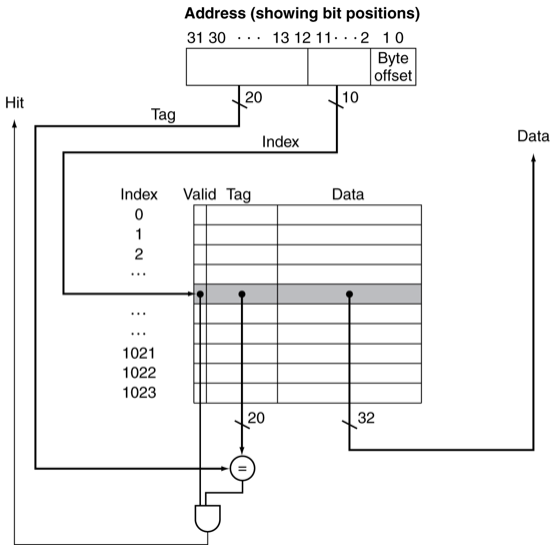
5.1.1 直接映射

[直接映射]

(1) 定义: 每个存储器地址对应 cache 中的唯一地址.



(2) 地址划分:



[**例**] 容量 8 块, 1 字/块, 直接映射.

初始状态

Index 索引	V 有效位	Tag 标记	Data 数据
000	N		
001	N		
010	N		
011	N		
100	N		
101	N		
110	N		
111	N		

Word addr	Binary addr	Hit/miss	Cache block
22	10 110	Miss	110

Index索引	V有效	Tag标志	Data数据
000	N		
001	N		
010	N		
011	N		
100	N		
101	N		
110	Y	10	Mem[10110]
111	N		

Word addr	Binary addr	Hit/miss	Cache block
26	11 010	Miss	010

Index	V	Tag	Data
000	N		
001	N		
010	Y	11	Mem[11010]
011	N		
100	N		
101	N		
110	Y	10	Mem[10110]
111	N		

Word addr	Binary addr	Hit/miss	Cache block
22	10 110	Hit	110
26	11 010	Hit	010

Index	V	Tag	Data
000	N		
001	N		
010	Y	11	Mem[11010]
011	N		
100	N		
101	N		
110	Y	10	Mem[10110]
111	N		

Word addr	Binary addr	Hit/miss	Cache block
16	10 000	Miss	000
3	00 011	Miss	011
16	10 000	Hit	000

Index	V	Tag	Data
000	Y	10	Mem[10000]
001	N		
010	Y	11	Mem[11010]
011	Y	00	Mem[00011]
100	N		
101	N		
110	Y	10	Mem[10110]
111	N		

Word addr	Binary addr	Hit/miss	Cache block
18	10 010	Miss	010

Index	V	Tag	Data
000	Y	10	Mem[10000]
001	N		
010	Y	10	Mem[10010]
011	Y	00	Mem[00011]
100	N		
101	N		
110	Y	10	Mem[10110]
111	N		

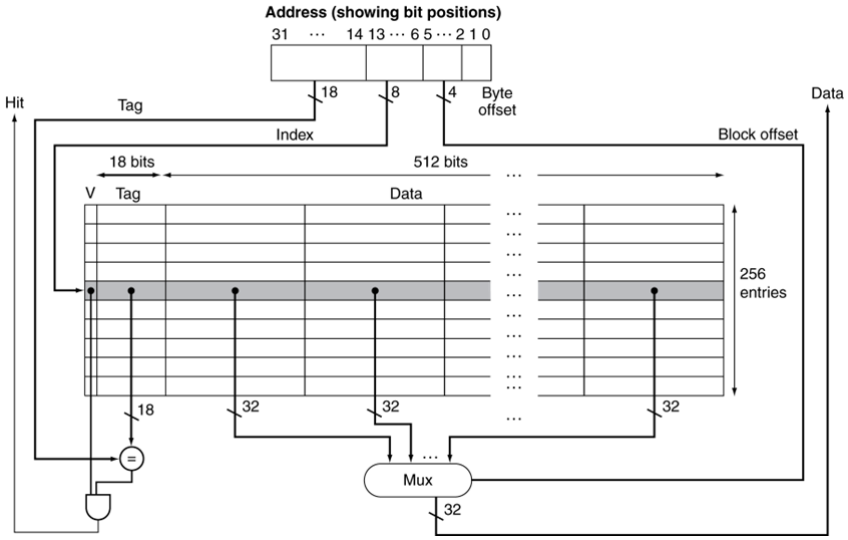
[**例**] 一个 cache 有 64 块, 每块 16 Bytes . 求字节地址 1200 被映射到 cache 的哪块.

[**解**] 主存块大小 = cache 块大小 = 16 Bytes .

$$\text{主存块号} = \left\lfloor \frac{\text{字节地址}}{\text{主存块大小}} \right\rfloor = \left\lfloor \frac{1200}{16} \right\rfloor = 75 ,$$

cache 块号 = 主存块号 mod cache 块数 $= 75 \bmod 64 = 11$.

[**FastMATH 处理器**]



[**cache 的性能**]

$$\text{存储器阻塞时钟周期数} = \frac{\text{存储器访存次数}}{\text{程序数}} \times \text{缺失率} \times \text{缺失代价} = \frac{\text{指令数}}{\text{程序数}} \times \frac{\text{缺失数}}{\text{指令}} \times \text{缺失代价}.$$

[**例**] 设 I-cache 缺失率 $= 2\%$, D-cache 缺失率 $= 4\%$, 缺失代价 $= 100$ 时钟周期, 处理器 $\mathrm{CPI} = 2$, load 和 store 占全部指令的 36%. 求: 配置一个从不发生缺失的理想 cache, 处理器速度快多少.

[**解**] 缺失时钟周期数:

① I-cache : $0.02 \times 100 \times I = 2I$.

② D-cache: $0.36 \times 0.04 \times 100 \times I = 1.44I$.

实际 $\mathrm{CPI} = 2 + 2 + 1.44 = 5.44$, 理想 CPU 是 $\dfrac{5.44}{2} = 2.72$ 倍.

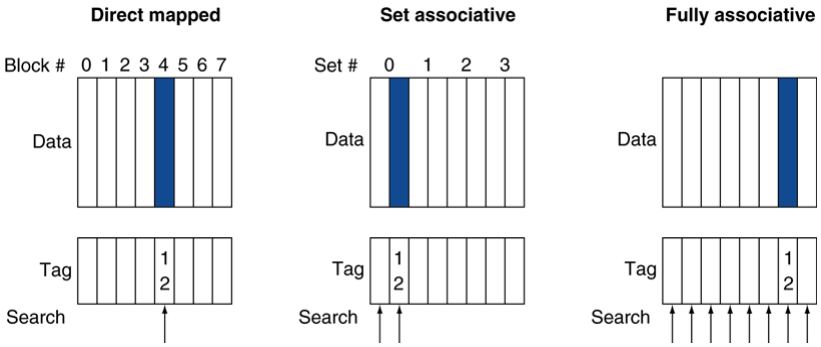
[**存储器的平均访问时间**] $\mathrm{AMAT} = \text{命中时间} + \text{缺失率} \times \text{缺失代价}$.

[**例**] CPU 时钟周期时间为 $1\ \mathrm{ns}$, 命中时间为 1 个时钟周期, 缺失代价为 20 个时钟周期, I-cache 的缺失率为 5%. 求 AMAT.

[**解**] $\mathrm{AMAT} = 1 + 0.05 \times 20 = 2\ \mathrm{ns}$, 即平均每条指令需 2 个时钟周期.

5.1.2 相联

[**相联**]



(1) 全相联:

① 一个块可被放置在 cache 中的任意位置.

② 需查找 cache 中的所有项.

③ 每个 cache 项都需比较器, 开销大.

(2) 多路组相联:

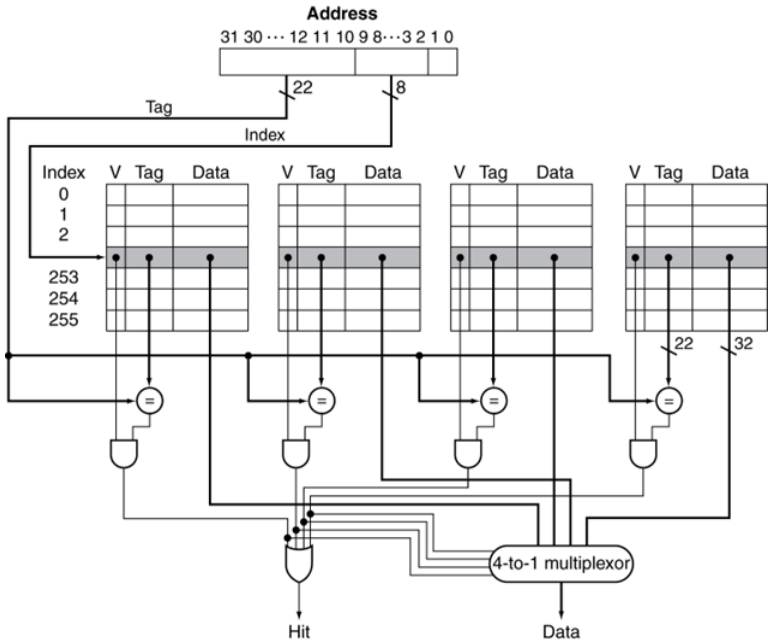
① 每个组包含 n 块.

② $\text{组号} = \text{块号} \bmod \text{组数}$.

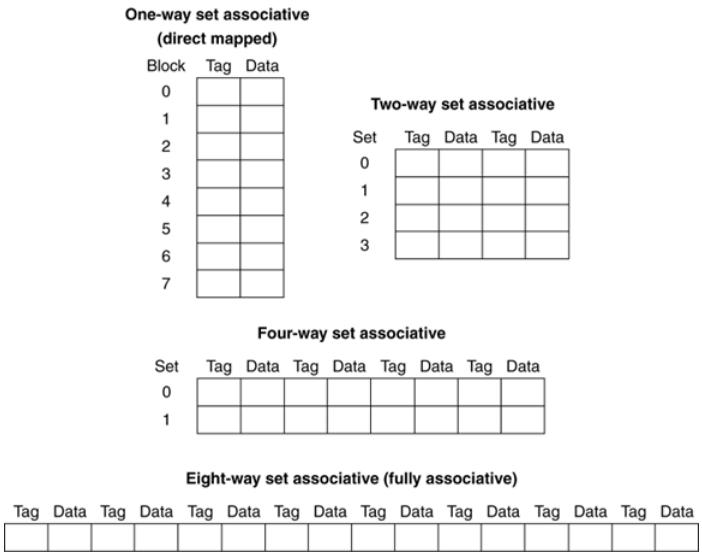
③ 可能需查找组中的所有块.

④ 只需 n 个比较器, 比直接映射开销少.

⑤ 结构:



(3) 相联度: 有 8 个块的 cache 配置为一路组相联(直接映射)、二路组相联、四路组相联、八路组相联(相联).



(4) 增加相联度可降低缺失率, 但收益递减.

[**例**] cache 含 4 块, 块访问顺序 0 , 8 , 0 , 6 , 8 . 采用直接映射、二路组相联、全相联的状态如下.

(1) 直接映射:

Block address	Cache index	Hit/miss	Cache content after access			
			0	1	2	3
0	0	miss	Mem[0]			
8	0	miss	Mem[8]			
0	0	miss	Mem[0]			
6	2	miss	Mem[0]		Mem[6]	
8	0	miss	Mem[8]		Mem[6]	

(2)

二路组相联

Block address	Cache index	Hit/miss	Cache content after access			
			Set 0		Set 1	
0	0	miss	Mem[0]			
8	0	miss	Mem[0]	Mem[8]		
0	0	hit	Mem[0]	Mem[8]		
6	0	miss	Mem[0]	Mem[6]		
8	0	miss	Mem[8]	Mem[6]		

全相联

Block address		Hit/miss	Cache content after access			
0		miss	Mem[0]			
8		miss	Mem[0]	Mem[8]		
0		hit	Mem[0]	Mem[8]		
6		miss	Mem[0]	Mem[8]	Mem[6]	
8		hit	Mem[0]	Mem[8]	Mem[6]	

[**替换策略**]

(1) 直接映射: 别无选择.

(2) 组相联:

① 选择存储非有效信息 (non-valid) 的块, 若无 non-valid 的块, 则任选一块.

② LRU (Least-Recently Used) : 选择最长时间未被使用的. 二路易实现, 四路可控制, 超过四路难控制.

③ 相联度较高时, 随机替换近似 LRU 的结果.

5.1.3 多级 cache

[**cache 缺失原因**, **3 C**]

(1) [**强制缺失**, **冷启动缺失**, **Compulsory Misses**] 第一次访问从未在 cache 中出现过的块的缺失.

(2) [**容量缺失**, **Capacity Misses**] cache 容量不足以容纳一个程序执行所需的所有块引起的缺失.

[**例**] 一个被替换的块随后又被访问.

(3) [**冲突缺失**, **Conflict Misses**] 在直接映射或组相联的 cache 中, 多个块竞争同个组时引起的缺失.

[**注**] 同容量的全相联 cache 不出现该情况.

[**cache 设计的折中**]

设计变化	对缺失率的影响	可能对性能的负面影响
增加 cache 容量	减少容量缺失, 降低缺失率	增加访问时间
提高相联度	减少冲突缺失, 降低缺失率	增加访问时间
增加块容量	因空间局部性, 对宽范围内变化的块大小 都能降低缺失率	① 增加缺失代价 ② 块过大时增加缺失率

[**多级 cache**]

(1) ① L1-cache 直连 CPU , 容量小, 速度快.

② L2-cache 处理 L1-cache 的失效, 容量较大, 速度较慢, 但仍快于主存.

③ 主存处理 L2-cache 的失效.

(2) ① L1-cache: 集中在最小化命中时间.

② L2-cache: 集中在低缺失率, 避免访问主存, 命中时间对整体性能影响较小.

(3) ① L1-cache 常比独立 cache 的容量小.

② L1-cache 的块大小常比 L2-cache 的块大小小.

[**例**] 设 CPU 基准 $\mathrm{CPI} = 1$, 时钟频率 $= 4\ \mathrm{GHz}$, 缺失率 / 指令 $= 2\%$, 主存访问时间 $= 100\ \mathrm{ns}$.

(1) 只有 L1-cache:

缺失代价 $= \dfrac{100\ \mathrm{ns}}{0.25\ \mathrm{ns}} = 400$ 时钟周期 , 有效 $\mathrm{CPI} = 1 + 0.02 \times 400 = 9$.

(2) 增加 L2 - cache , 访问时间 $5\ \mathrm{ns}$, 对主存的整体失效率 $= 0.5\%$.

① L1 失效, L2 命中: 缺失代价 $= \dfrac{5\ \mathrm{ns}}{0.25\ \mathrm{ns}} = 20$ 时钟周期 .

② L1 失效, L2 失效: 额外缺失代价 $= 400$ 时钟周期 .

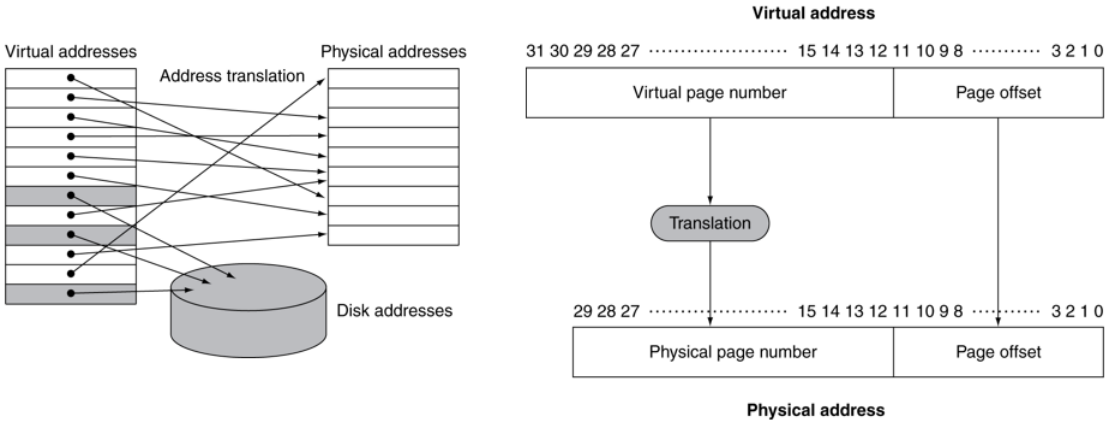
$\mathrm{CPI} = 1 + 0.02 \times 20 + 0.005 \times 400 = 3.4$, 性能比 $= \dfrac{9}{3.4} \approx 2.6$.

5.2 虚拟存储器

5.2.1 页表

[**地址映射**]

(1) 定义: 固定页大小, 虚页号到物理页号的转换.



(2) 若缺页, 需从磁盘上取回该页, 消耗几百万时钟周期, 由 OS 代码管理.

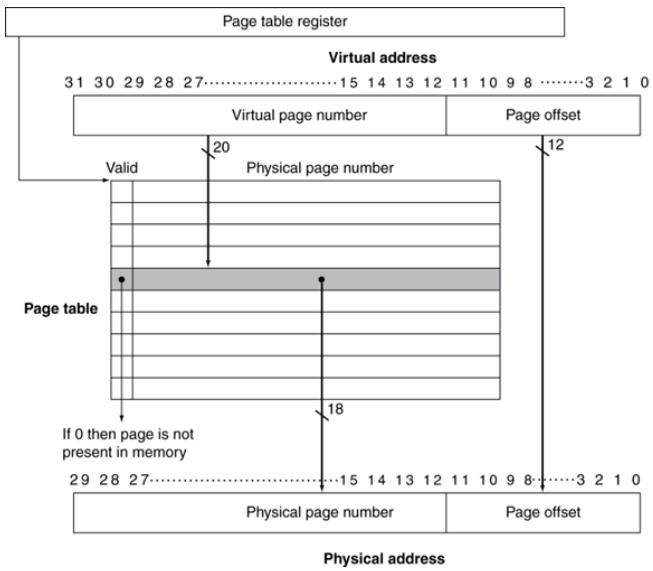
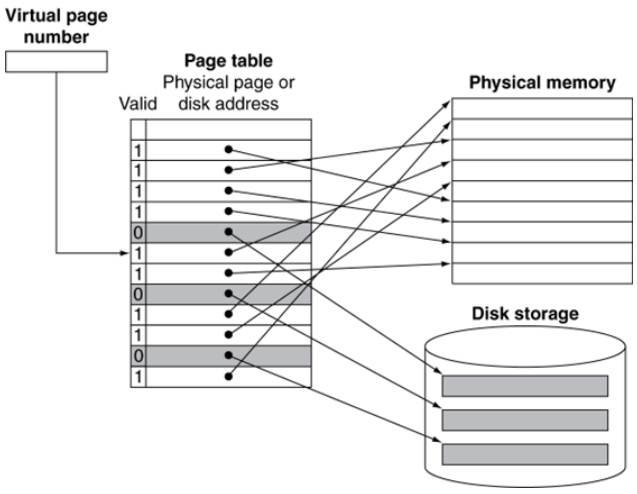
(3) 降低缺页率:

① 存储器中的页按全相联放置.

② 智能替换算法.

[**页表**] PTE(Page Table Entry).

(1)

页表转换	页表到存储的映射
	

(2) 用 LRU 替换算法降低缺页率.

① PTE 的引用位周期地被 OS 清零.

② 某页被访问时, PTE 中引用位 (aka 使用位) 置 1 .

③ 某页引用位为 0 说明近期未使用.

(3) 磁盘写入花费几百万时钟周期.

① 成块写入, 不单独写某地址.

② 写 write-through 在实际应用中不现实, 故采用 write-back.

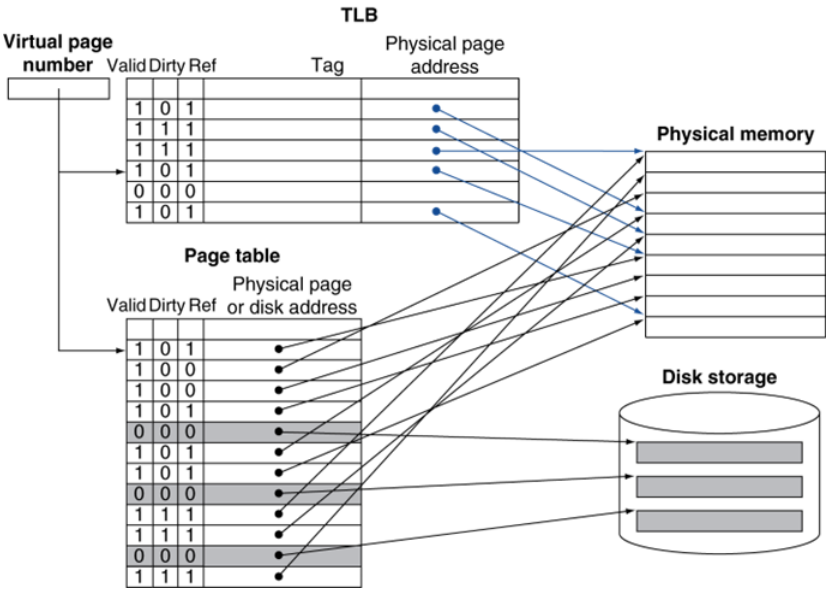
5.2.2 TLB

[**快表**, **地址变换高速缓存**, **Translation Look-aside Buffer**, **TLB**]

(1) 地址映射需额外的访存开销: 先访问 PTE , 再访问实际内存地址.

(2) 提高访存性能的关键是依靠页表的访问局部性. 现代 CPU 用一个特殊的 cache 跟踪近期使用的地址变换, 即 TLB .

(3) 用 TLB 进行快速地址转换:



[**TLB 缺失**]

(1) TLB 缺失情况:

① 页存在, 但 PTE 不在 TLB .

② 页不存在.

(2) TLB 缺失处理: 处理器从主存拷贝 PTE 到 TLB , 随后重启指令. 若不存在该页, 则发生缺页.

① 若页在主存, 从主存加载 PTE 后重试.

i) 硬件: 用更复杂的页表结构.

ii) 软件: 用更先进的替换算法.

② 若页不在主存, 即缺页, 则 OS 提取页并更新页表, 后重新开始执行缺页的指令.

(3) 缺页 (Page Fault, PF) 处理:

① 用发生缺失的虚拟地址查找 PTE .

② 定位磁盘上的页, 选择要替换的页, 若页状态为 dirty , 则先写到磁盘上.

③ 将页读到主存并更新页表.

④ 重新运行进程, 从发生缺页的指令开始重新运行指令.

[**例**] 设系统采用 4 KB 的页, 一个 4 项的全相联 TLB , 使用 LRU . 若必须从磁盘取回页, 则增加下一次能取的最大页数. 下面是该系统可见的虚拟地址流、 TLB 和页表的初始状态. 依次访问虚拟地址 4699 , 2227 , 13916 , 34587 , 48870 , 12608 , 49925 .

TLB 初始状态

有效位	标记位	物理页号
1	11	12
1	7	4
1	3	6
0	4	9

页表初始状态

有效位	物理页/硬盘上
1	5
0	硬盘
0	硬盘
1	6
1	9
1	11
0	硬盘
1	4
0	硬盘
0	硬盘
1	3
1	12

$$\text{页大小} = 4 \times 1024 \text{ Bytes} = 4096 \text{ Bytes} .$$

(1) 访问 4669 , $\text{tag} = \left\lfloor \dfrac{4669}{4096} \right\rfloor = 1$, 则虚拟页号为 1 . TLB miss, PT miss, 发生 PF .

因页表有空项, 则内存有空闲页, 此时内存的最大物理页为 12 , 则下一个空闲页为 13 .

TLB 有无效的项, 替换为 $\text{tag} = 1$ 对应物理页 13 .

有效位	标记位	物理页号
1	11	12
1	7	4
1	3	6
0 → 1	4 → 1	9 → 13

有效位	物理页/硬盘上
1	5
0 → 1	硬盘 → 13
0	硬盘
1	6
1	9
1	11
0	硬盘
1	4
0	硬盘
0	硬盘
1	3
1	12

(2) 访问 2227 , $\text{虚拟页号} = \text{tag} = \left\lfloor \dfrac{2227}{4096} \right\rfloor = 0$. TLB miss, PT hit. 按 LRU 替换 TLB .

替换

有效位	标记位	物理页号
1	11	12
1	7	4
1	3	6
1	1	13
1	0	5

有效位	物理页/硬盘上
1	5
1	13
0	硬盘
1	6
1	9
1	11
0	硬盘
1	4
0	硬盘
0	硬盘
1	3
1	12

(3) 访问 13916，虚拟页号 $=$ tag $=\left\lfloor \dfrac{13916}{4096} \right\rfloor = 3$，TLB hit. 按 LRU 更新 TLB 顺序.

有效位	标记位	物理页号
1	7	4
1	3	6
1	1	13
1	0	5
1	3	6

有效位	物理页/硬盘上
1	5
1	13
0	硬盘
1	6
1	9
1	11
0	硬盘
1	4
0	硬盘
0	硬盘
1	3
1	12

(4) 访问 34587，虚拟页号 $=$ tag $=\left\lfloor \dfrac{34587}{4096} \right\rfloor = 8$，TLB miss, PT miss, 发生 PF .

下一个空闲页为 14 . 按 LRU 替换 TLB .

有效位	标记位	物理页号
1	7	4
1	1	13
1	0	5
1	3	6
1	8	14

有效位	物理页/硬盘上
1	5
1	13
0	硬盘
1	6
1	9
1	11
0	硬盘
1	4
0-> 1	硬盘 -> 14
0	硬盘
1	3
1	12

(5) 访问 48870，虚拟页号 $=$ tag $=\left\lfloor \dfrac{48870}{4096} \right\rfloor = 11$，TLB miss, PT hit. 按 LRU 替换 TLB .

有效位	标记位	物理页号
1	1	13
1	0	5
1	3	6
1	8	14
1	11	12

有效位	物理页/硬盘上
1	5
1	13
0	硬盘
1	6
1	9
1	11
0	硬盘
1	4
1	14
0	硬盘
1	3
1	12

(6) 访问 12608，虚拟页号 $=$ tag $=\left\lfloor \dfrac{12608}{4096} \right\rfloor = 3$，TLB hit. 按 LRU 更新 TLB .

有效位	标记位	物理页号
1	0	5
1	3	6
1	8	14
1	11	12
1	3	6

有效位	物理页/硬盘上
1	5
1	13
0	硬盘
1	6
1	9
1	11
0	硬盘
1	4
1	14
0	硬盘
1	3
1	12

(7) 访问 49925 , 虚拟页号 $=$ tag $=\left\lfloor\dfrac{49925}{4096}\right\rfloor=12$, TLB miss, PT miss, 发生 PF .

下一个空闲页为 15 . 按 LRU 替换 TLB .

有效位	标记位	物理页号
1	0	5
1	8	14
1	11	12
1	3	6
1	12	15

有效位	物理页/硬盘上
1	5
1	13
0	硬盘
1	6
1	9
1	11
0	硬盘
1	4
1	14
0	硬盘
1	3
1	12
1	15