ITP 30002 Operating System

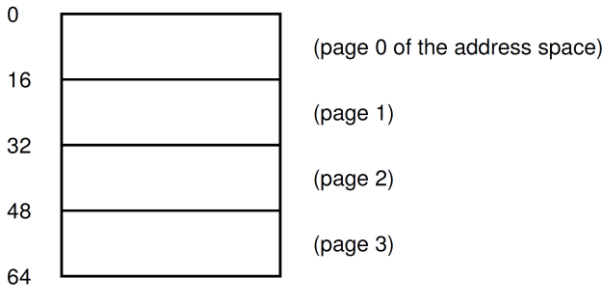
Paging

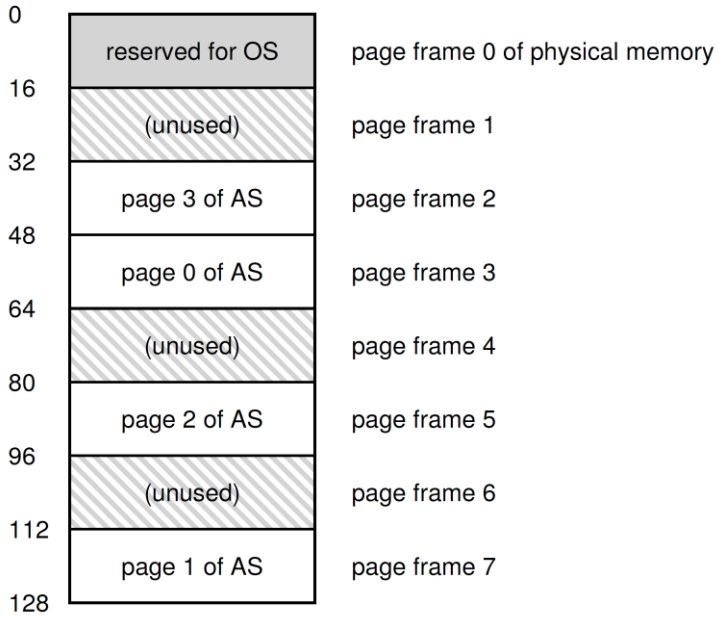
OSTEP Chapters 18 and 19

Shin Hong

Paging

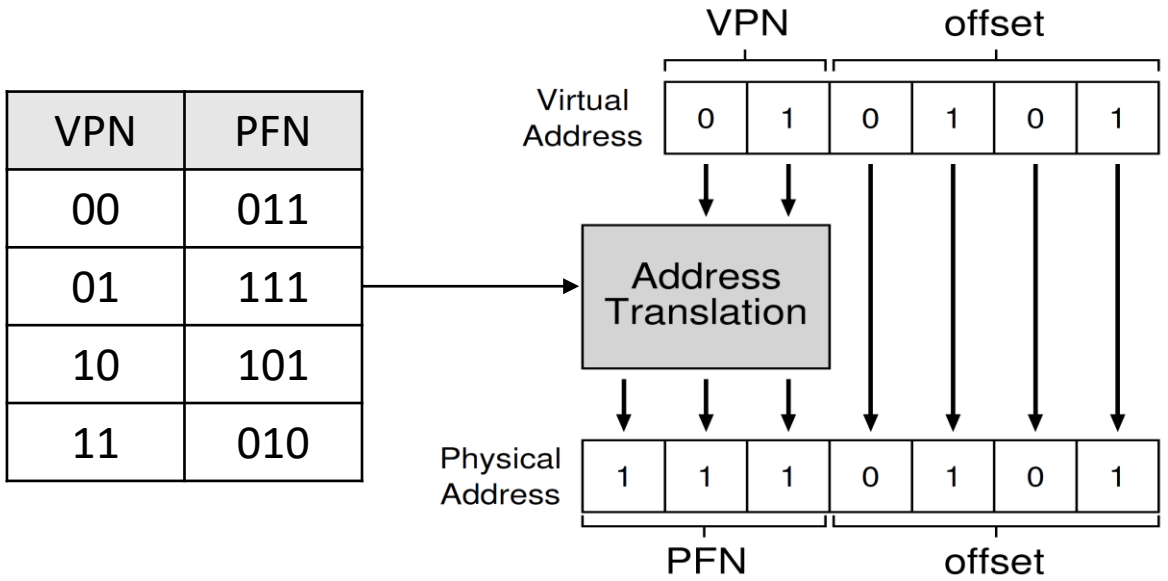
- allocate a memory resource to a process as a set of small fixed-size pieces
  - divide an address space into **pages** and divide a physical memory space into **frames** such that the sizes of a page and a frame are the same
  - can resolve both internal and external fragementation problems if the maintenance overhead is manageable
- ex. 16-byte pages/frames





Page Table and Address Translation

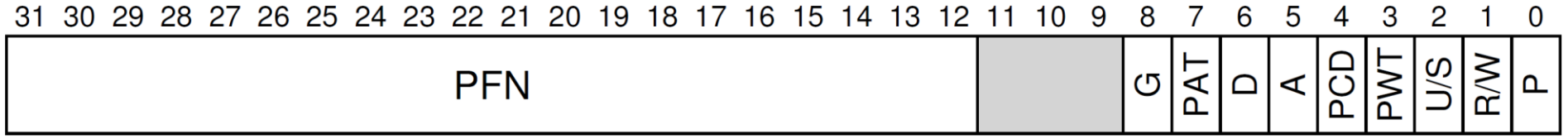
- there must be a page table which records to which frame a page of a process is assigned
  - structured as per-process page table, or inverted page table
- to translate a virtual address, we have to split it into the VPN and the offset, and replace VPN with the corresponding PFN
  - ex. a 64-byte address space in a 128-byte physical memory



Page Table Entry

- valid bit: whether or not the page is in use
- permission bits: read-only, read-write, supermode-only, etc.
- present bit: whether the page is in physical memory or on disk
- dirty bit: whether the page was modified since it was brought into memory
- reference bit: whether the page has been recently accessed

- Example of 32-bit x86



Page Table Size

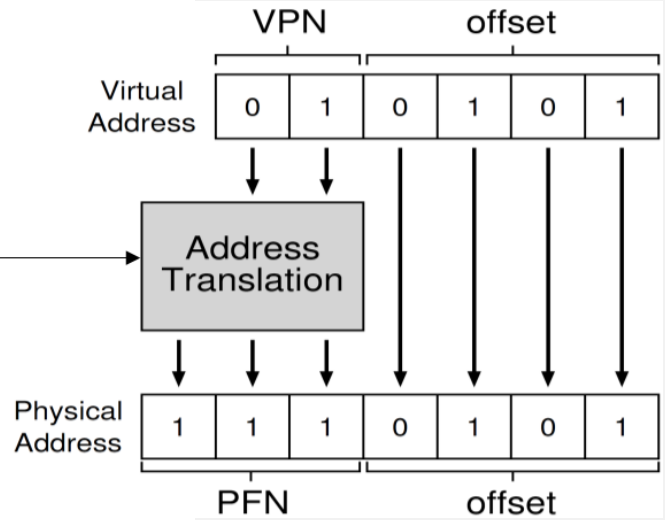
- page tables take a large amount of memory
  - example
    - 4 KB page/frame in a 32-bit address space and a 4 GB memory
    - 20-bits VPN, 2^{20} page table entries
    - 4 bytes for each entry, thus, 4 MB (=2^{22} bytes) for a page table
    - 400 MB for 100 processes

- a page table should be resided in a main memory since MMU cannot accommodate a full page table
  - every translation step requires at least one memory access

Accessing Memory with Paging


```
1  // Extract the VPN from the virtual address
2  VPN = (VirtualAddress & VPN_MASK) >> SHIFT
3
4  // Form the address of the page-table entry (PTE)
5  PTEAddr = PTBR + (VPN * sizeof(PTE))
6
7  // Fetch the PTE
8  PTE = AccessMemory(PTEAddr)
9
10 // Check if process can access the page
11 if (PTE.Valid == False)
12     RaiseException(SEGMENTATION_FAULT)
13 else if (CanAccess(PTE.ProtectBits) == False)
14     RaiseException(PROTECTION_FAULT)
15 else
16     // Access is OK: form physical address and fetch it
17     offset   = VirtualAddress & OFFSET_MASK
18     PhysAddr = (PTE.PFN << PFN_SHIFT) | offset
19     Register = AccessMemory(PhysAddr)
```

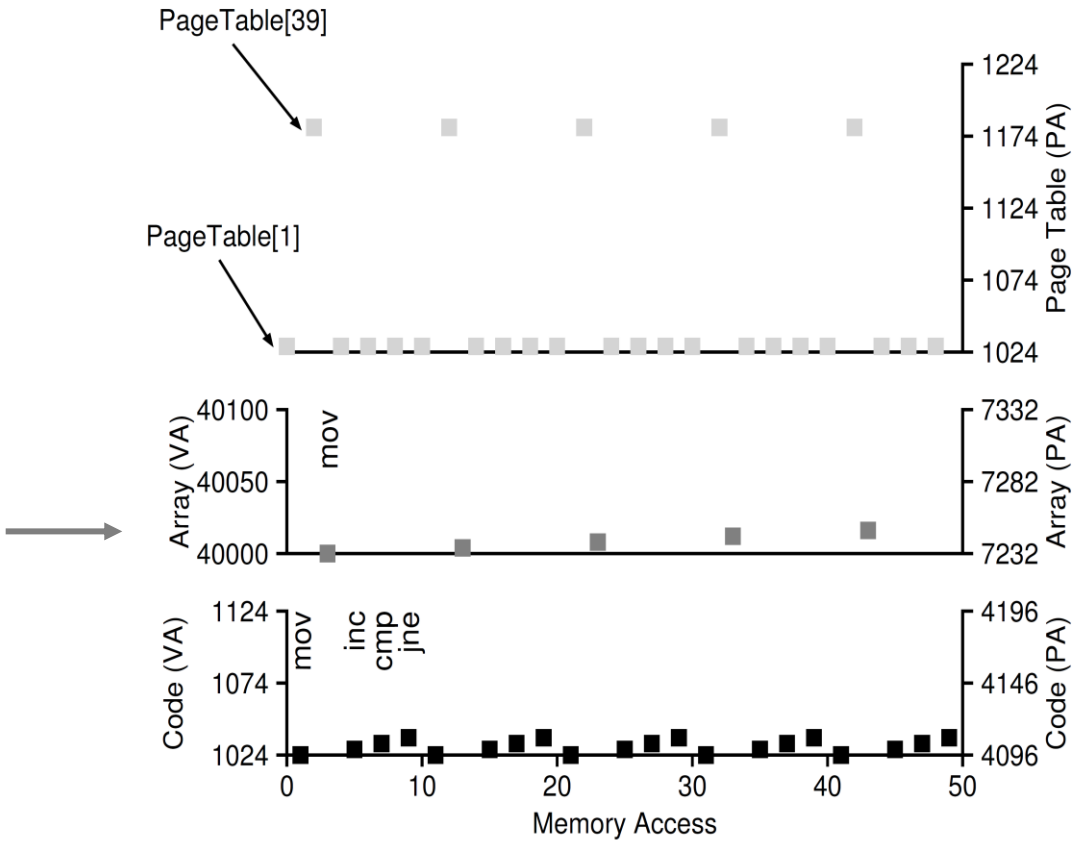
VPN	PFN
00	011
01	111
10	101
11	010



Ex. Memory Trace


```
int array[1000];
...
for (i = 0; i < 1000; i++)
    array[i] = 0;
```

```
1024 movl $0x0,(%edi,%eax,4)
1028 incl %eax
1032 cmpl $0x03e8,%eax
1036 jne  0x1024
```

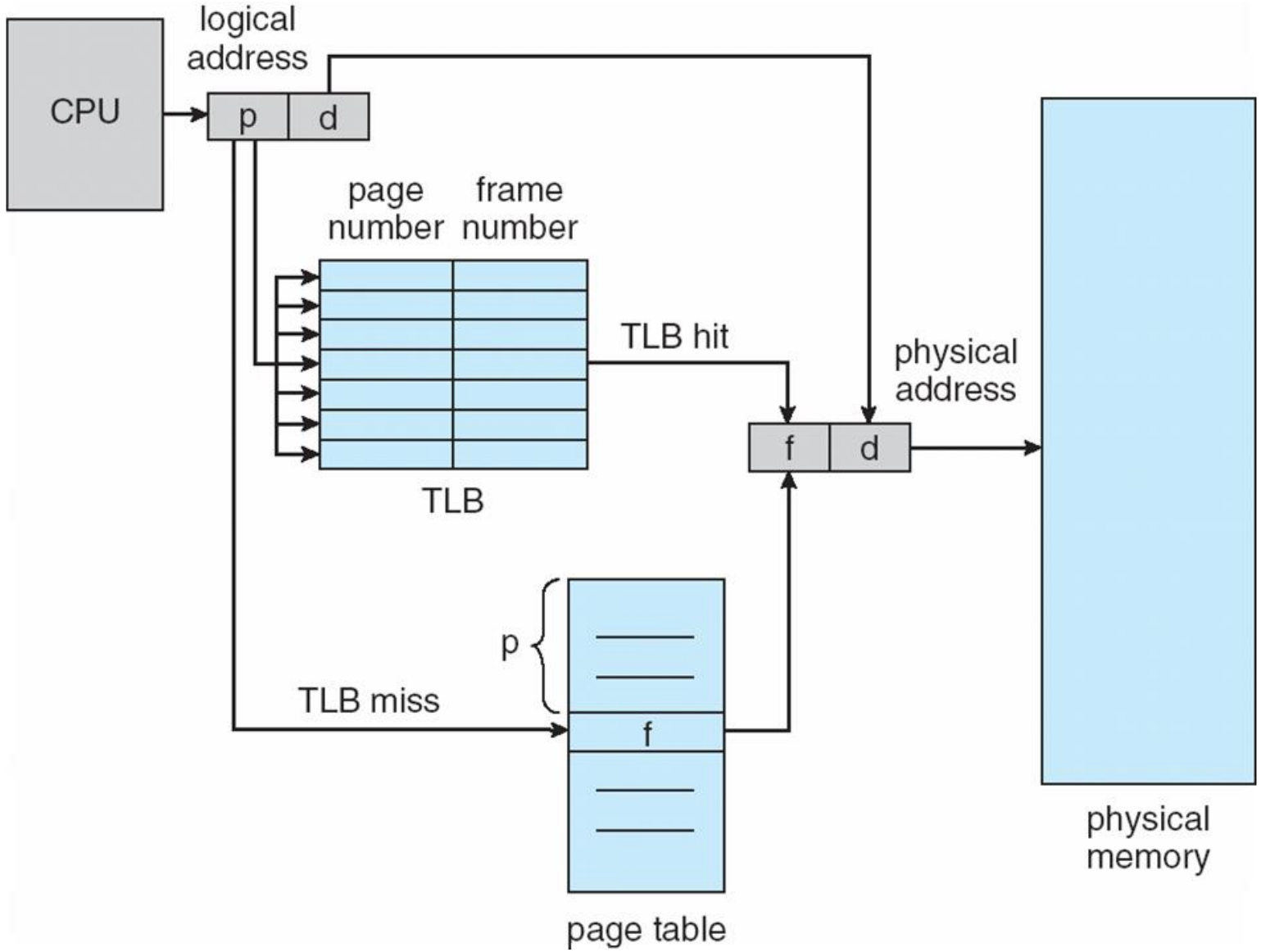


Translation-lookaside Buffer (TLB)

- A TLB is a part of MMU working as a cache of a page table
  - upon a memory access request, the computer architecture first checks the TLB to see if it has page translation information for the corresponding VPN
  - address-translation cache

- Steps for translating a virtual address with TLB
  - extract the VPN from a virtual address
  - check if the entry for the VPN is found in the TLB
    - if there exists, get the PFN from the TLB (i.e., TLB hit)
    - otherwise (i.e., TLB miss)
      - reference the page table to get the PFN
      - update the TLB with the VPN and the PFN
      - repeat from the beginning

Paging Hardware with TLB



Example

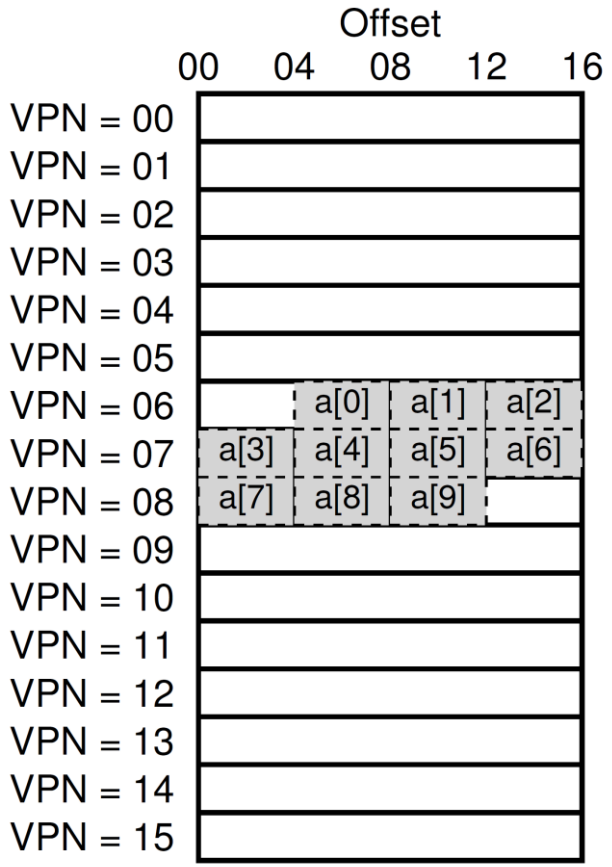

```
int sum = 0;
for (i = 0; i < 10; i++) {
    sum += a[i];
}
```

read 100 // a[0]
TLB miss
Update TLB

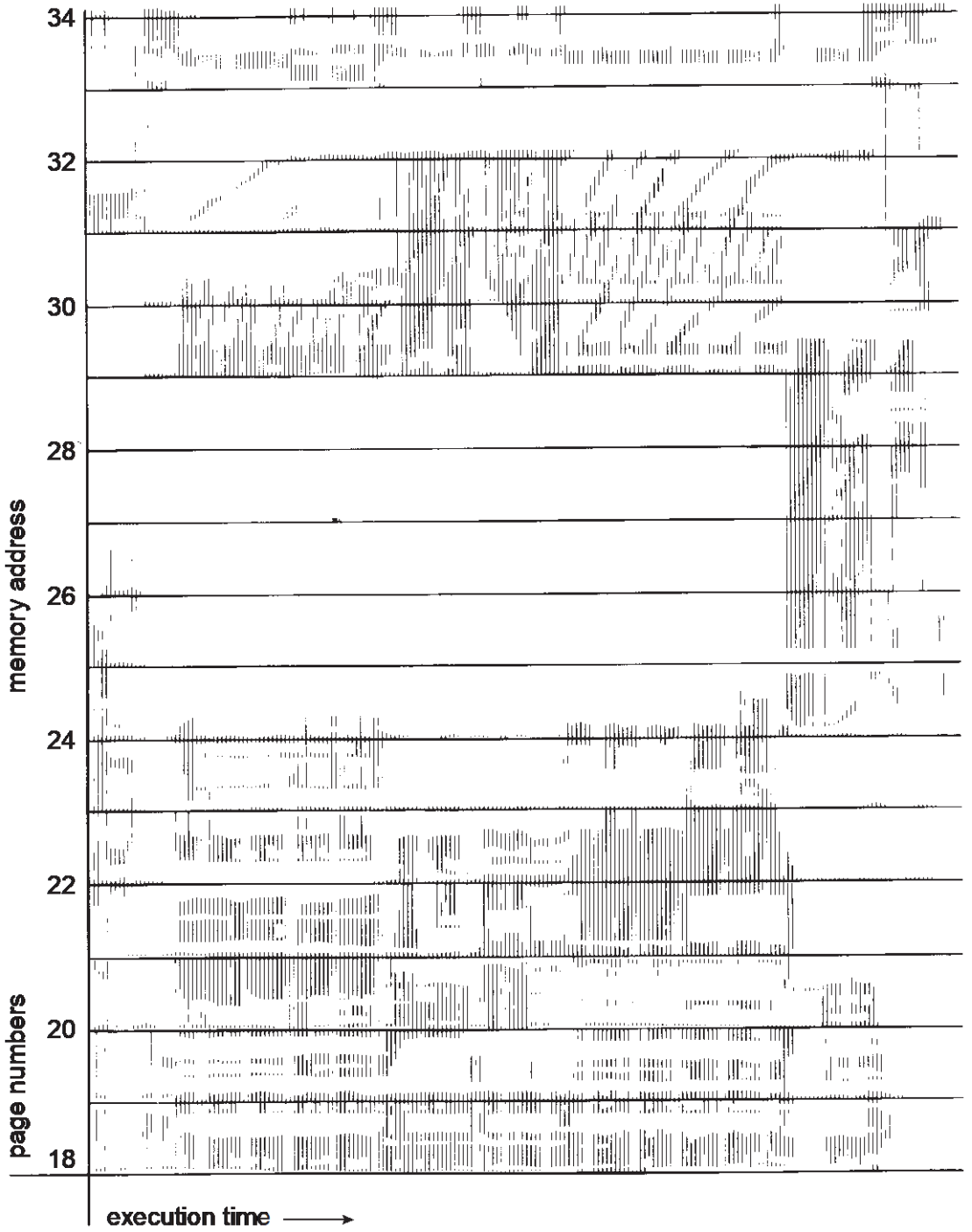
read 101 // a[1]
TLB hit

read 102 // a[2]
TLB hit

read 103 // a[3]
TLB miss
Update TLB



Memory Reference Pattern



Locality

- A TLB can save memory accesses because memory accesses of an application program tend to have **temporal locality** and **spatial locality**
  - **temporal locality**: if a program accessed a memory at address *x*, it will likely access *x* again in near future.
  - **spatial locality**: if a program accesses a memory at address *x*, it will likely access *x* + *d* in near future (where d is a small number)

- Trade off between TLB size and TLB access speed
  - The bigger the size of a TLB is, the higher the hit ratio is
  - The bigger the size of a TLB is, the longer the look-up time takes