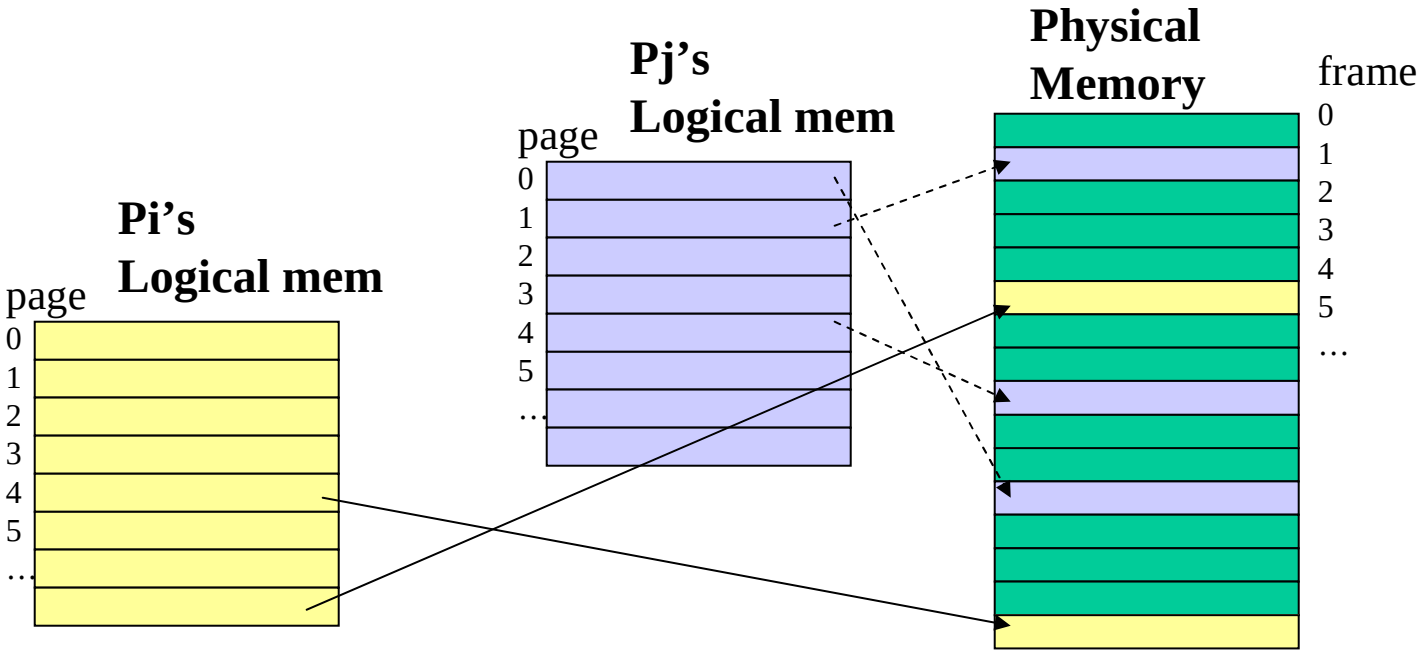
# Paging

Problem: if logical address space must be in contiguous physical memory space:

- decrease memory utilization
- external fragmentation
- complicated memory management algorithms

Solution: let Pi's address space be loaded into non-contiguous memory chunks

Idea: * break logical address space into fixed-size chunks (PAGES)
* break physical memory into same sized chunks (FRAMES)
* any PAGE of logical address space can be loaded into any FRAME

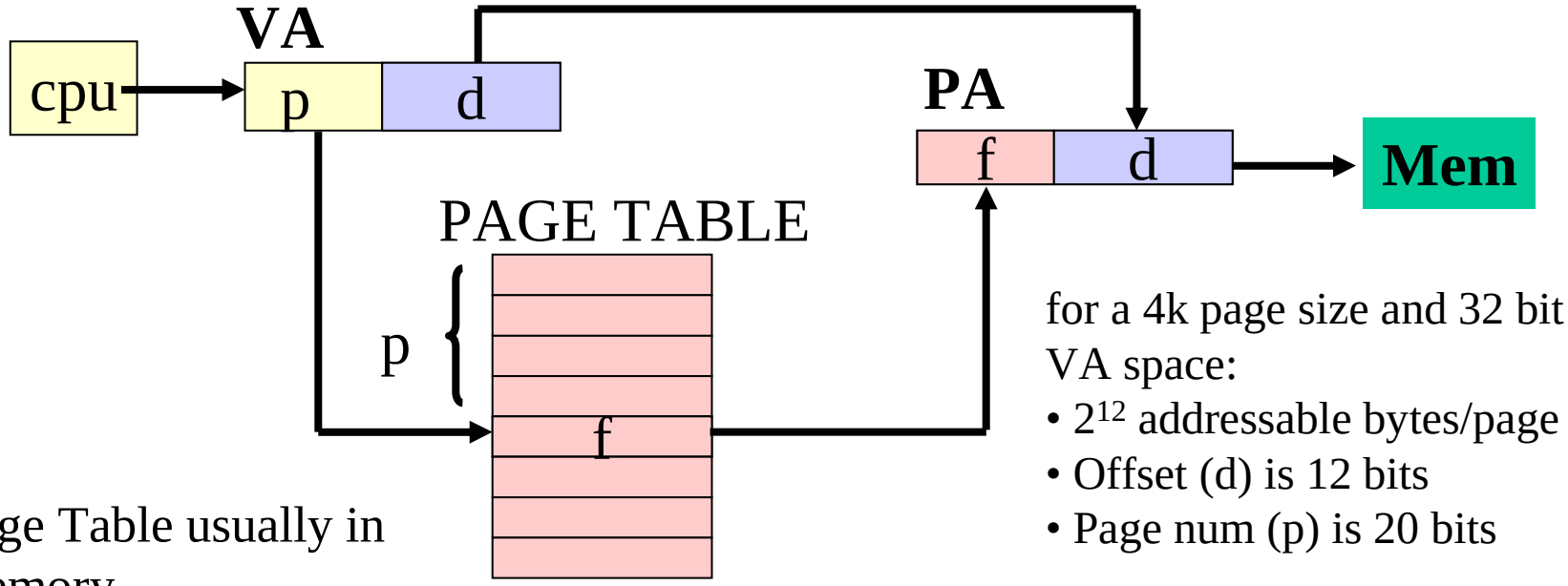


# Paging

Problem: how to find frame in to which logical page j of a process is (its not just at some offset in RAM from Pi's start)

Solution: "Add a level of indirection"
PAGE TABLE where entries map logical pages of Pi to physical page frames

Virtual Address consists of page number (p) and offset into page (d)
Physical Address consists of frame number (f) and offset into page (d)




Page Table usually in memory

for a 4k page size and 32 bit VA space:
- $2^{12}$ addressable bytes/page
- Offset (d) is 12 bits
- Page num (p) is 20 bits

•We need some HW support for paging

- lots of differences across architectures

Usually: some registers for paging (PTBR, PTLR)

privileged instructions for saving/restoring

dispatcher saves/restores these to PCB on CXS

Some systems OS does page table lookup (SPARC)

Some systems HW does (x86)

Some systems Page Table in HW regs (DEC-PDP11)

Problem: Now every memory access takes two memory accesses

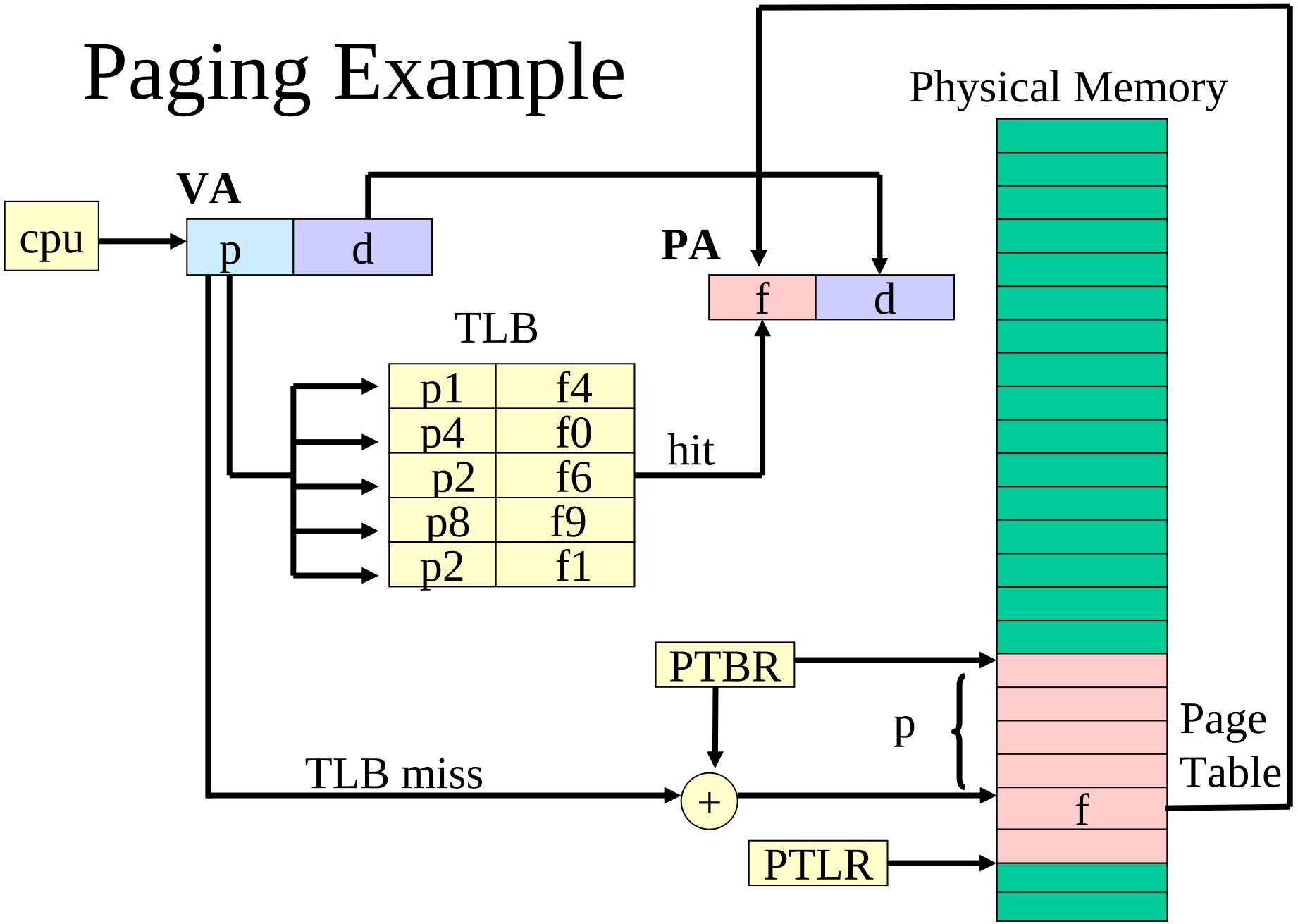
1 to read PT entry to get frame #

1 to read byte of memory in frame *f* at offset *d*

Solution: Special HW: keep a cache of address translations in super fast memory

TLB- Translation Look aside Buffer

# Paging Example



# Effective Memory Access Time

- TLB Hit Rate (H): % time mapping found in TLB
- Effective Access Time:

  [ (H)(TLB access time + mem access time)

  + (1-H)(TLB access + PT access + mem access)]

- Example: mem access: 100ns, TLB: 20ns, hit rate: 80%

  Effective Access Time = (.8)(120) + (.2)(220) = 140 ns

  ➔ 40% slowdown in mem. access time due to paging

  w/ 98% hit rate, effective access time is122 ns
- with 64-128 entries, TLB hit rate ~98%

  <u>locality of reference</u>

- Need:
  - TLB Replacement Policy
    - on TLB miss new mapping should go in TLB
    - which mapping to replace (kick out of TLB)?
  - Flush TLB on Context Switch
    - TLB logical to physical address mappings are only valid for currently running process
    - Protect processes from each other
- Paging:
  - Separate logical and physical view of memory:
    - logical view: one large contiguous set of addressable bytes
    - Physical view: set of page-sized frames that can hold any page of a processes memory
  - No External Fragmentation
  - Level of Indirection
    - Each process has a page table mapping its VAs to Pas
    - Need special HW to make paging fast