What if Virtual Address Space is Big?

- 32 bit addresses can address 2^{32} bytes of memory
- Page Size of 4KB
  - low order 12 bits of VA for offset within a page
  - leaves 20 bits for page number ➔ 2^{20} pages VA space
  - 2^{20} page table entries for full VA space
    - If each PT entry is 4bytes need:

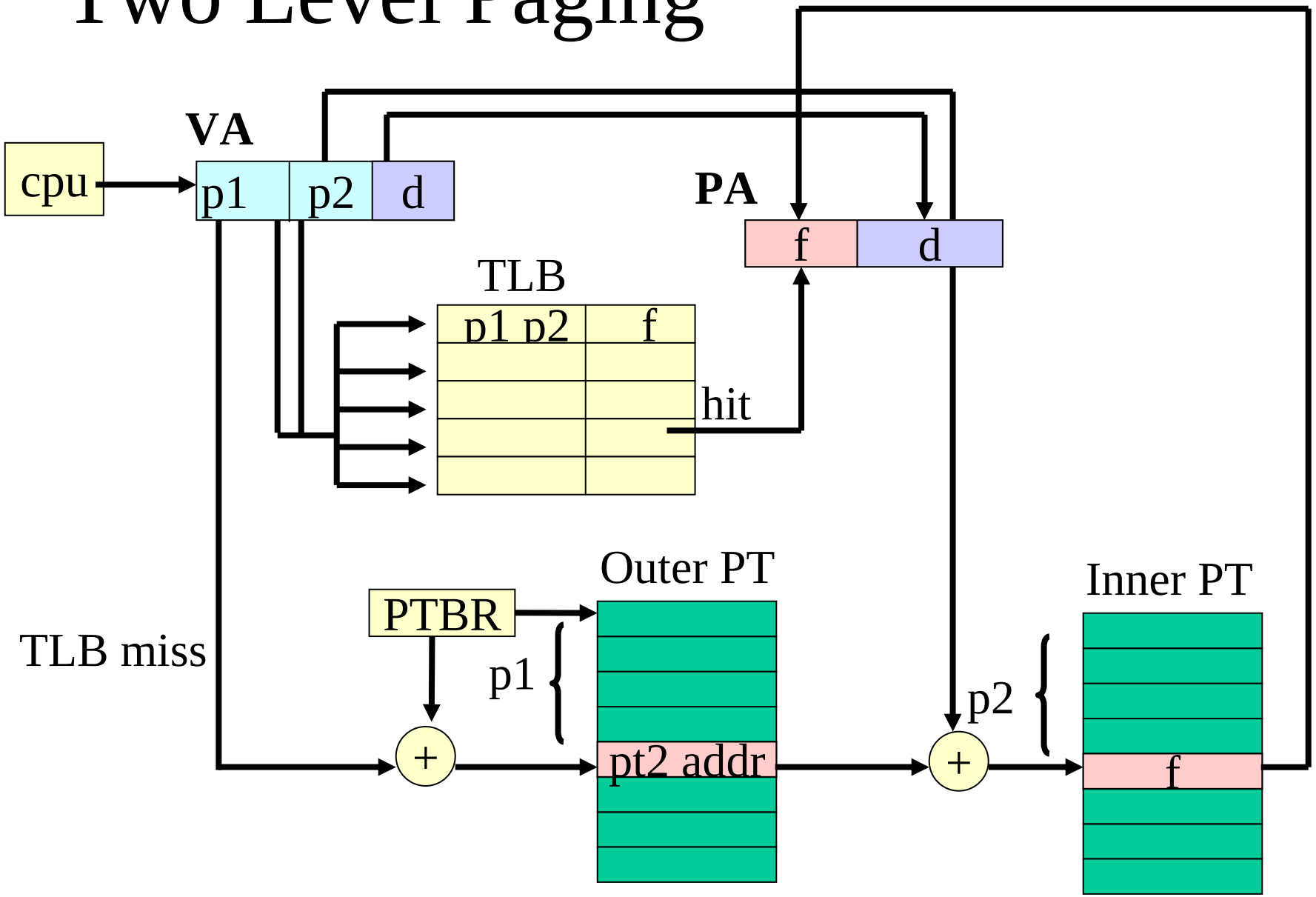
      2^{22} bytes of contiguous memory space for Page Table

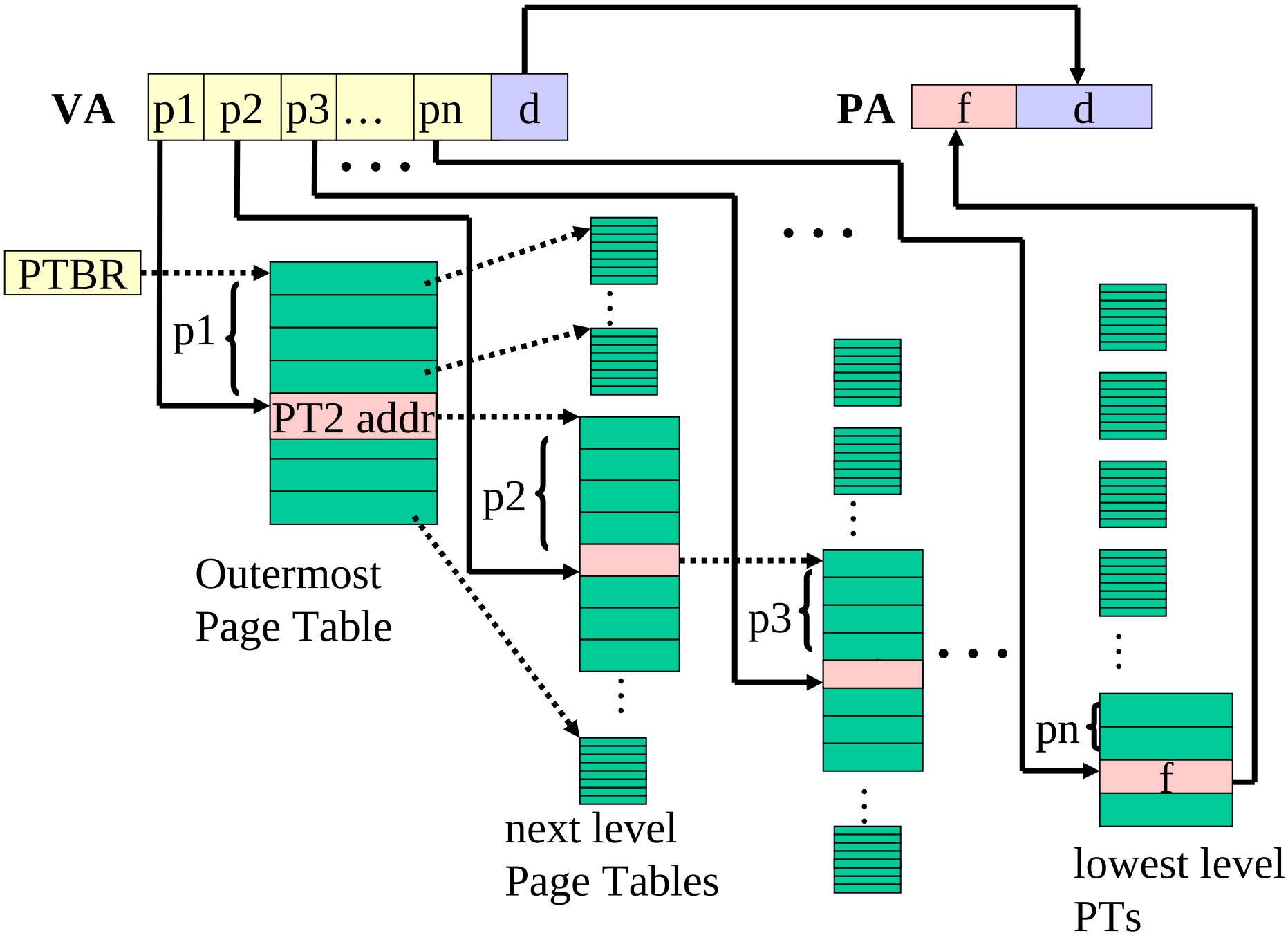
      (2^{10} contiguous pages of memory for Page Table)

      ➔ have the contiguous memory allocation problem again
- Solution: Page the Page Table
    - Allow it to live in non-contiguous memory
    - Don't need to have whole PT resident

    ➔ now we need a page table for the page table

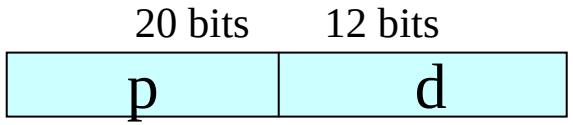
Two Level Paging




VA
p1
p2
p3
…
pn
d
PA
f
d
PTBR
p1
PT2 addr
p2
p3
pn
f
Outermost
Page Table
next level
Page Tables
lowest level
PTs

How Many Levels of PT?

- Depends on:
  - Virtual address space size
  - Page size
  - PT entry size
  - Number of contiguous pages in the outer most PT
    (assume only 1 unless otherwise specified)
- Example:
  - 4k (2^{12} bytes) page size, 32 bit VA, 8 byte PT entries
  - VA:

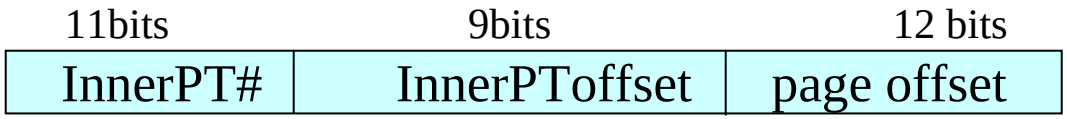


  - Need 2^{20} PT entries each 8 bytes long (2^{23} bytes of PT space)
  - Can fit $2^{12}/2^{3}$ (2^{9}) PT entries per PT page
    ➔ need $2^{20}/2^{9}$ (2^{11}) pages of inner-most page tables

Example Continued

Can fit $2^{12}/2^3$ (2^9) entries per page (need 9 bits for offset into inner PT page)

➔ need $2^{20}/2^9$ (2^{11}) pages of inner-most page table pages



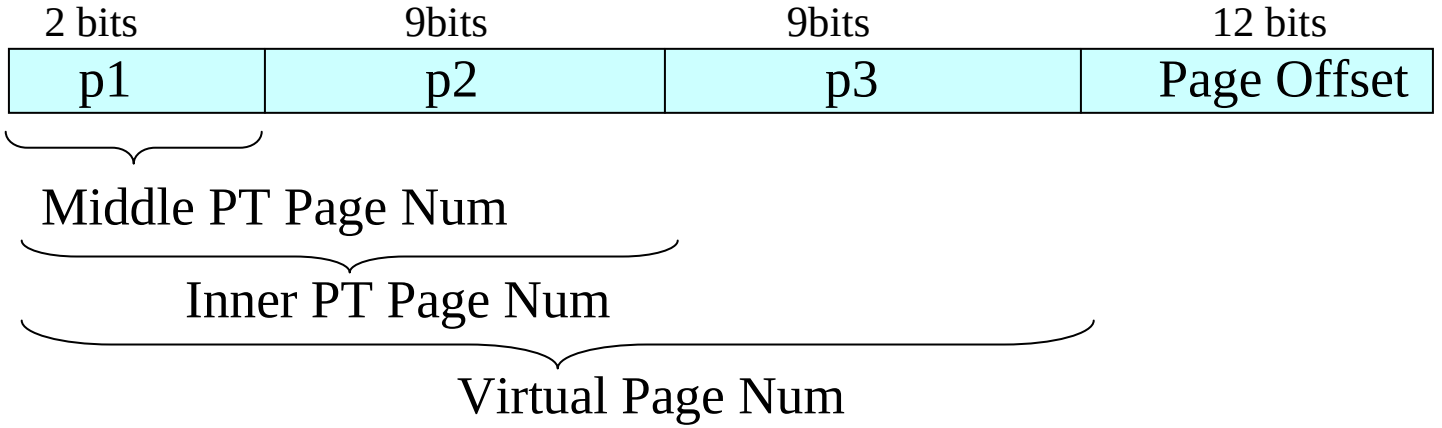
Outer PT entries (entries point to inner PT pages) don't fit on single page

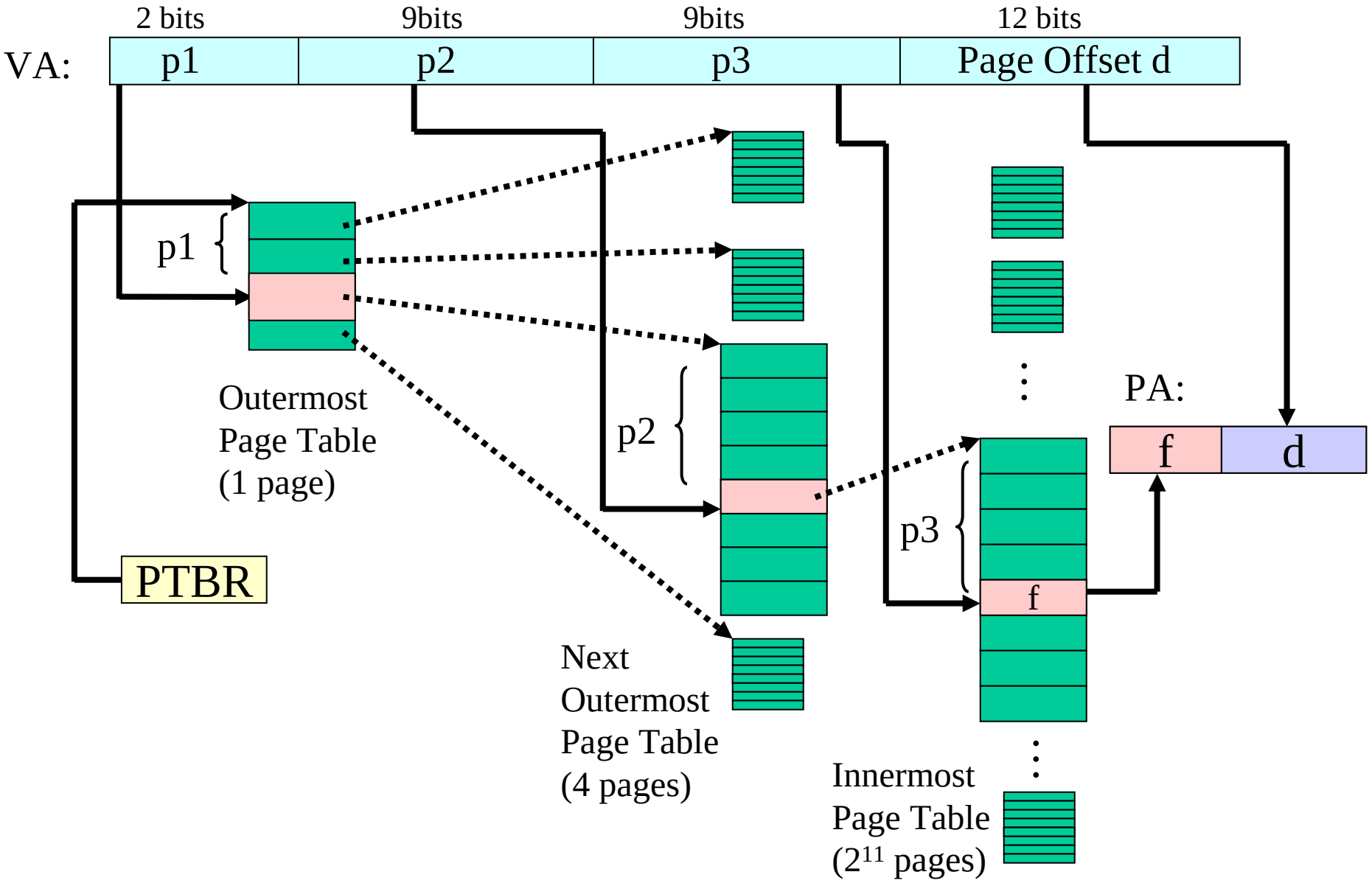
➔ we need another level of page tables (page the outer PT)

Number of outer level PT pages to refer to 2^{11} pages of Inner PT:

can fit 2^9 PT entries per page, so need 2^2 pages of 2^{nd} outer-level PT

The 2^{nd} outer-level PT entries fit on a single second-level outer PT page




2 bits
9bits
9bits
12 bits
VA:
p1
p2
p3
Page Offset d
p1
Outermost
Page Table
(1 page)
PTBR
p2
Next
Outermost
Page Table
(4 pages)
p3
f
Innermost
Page Table
(2^11 pages)
PA:
f
d

Effective Memory Access Time

Depends on Number of Levels of Page Tables:

Effective Access Time:

[(H)(TLB access time + mem access time)
+ (1-H)(TLB access + PTLevels(PT access) + mem access)]

Example: memory access time 100ns
TLB access time 20 ns
TLB hit rate = 80%
Number PT levels = 3

Effective Access Time =

(.8)(120) + (.2)(20 + 3(100) + 100) = 204 ns

➔ more than twice as slow due to paging

What if VA space is really big?

- 64 bit architectures: 2^{64} bytes of VA space

  ~ 7 levels of Page Tables

  Effective Access Time is way too slow

  Multiple Pi's PTs take up way too much of mem

- Solution: Inverted Page Tables
  - Idea: instead of mapping from VA to frame #
    keep mappings of frame# to VA
  - PA space much smaller than VA space
    inverted PT small (one entry per page FRAME)
  - Single, global PT for all processes in the system