Paging

virtual addr → phys. addr

virt.

Pages

\[ \text{A} \]
\[ \text{B} \]
\[ \text{C} \]
\[ \text{D} \]

Mem

\[ \text{B} \]
\[ \text{D} \]
\[ \text{A} \]
\[ \text{C} \]

frames

phys
page size: 1kB
10 bit addr.

0...2^{13} - 1
Page table ← process

Page ← frame

virt addr ← offset

Page table also in mem.
Free page list
- shared by processes
- bit map (one bit per frame)

Page table - multiple memory access
TLB - hardware for paging
- translation look aside buffer
- cache of page table
Cache implementation

direct mapped

hash

6000 1 data

set-associative memory

data v data, addr v data

6000 1 7000 1

ad
ad

fully-associative
(content addressable)
Cache operation

Is the address in cache?

yes - cache hit
bring data to CPU

no - cache miss
get data from memory
put it in cache
(remove other?)
32 bit logical address
page/frame size 4 KB
page number 20 bits
20 page table entries
$2^{20} = 4 \text{ MB page table}$
Hierarchical page tables

page #

offset

page numbers: 1, 2, ...

fields: f, d
Other types

- Hashed page table
- Inverted page table (frame table)