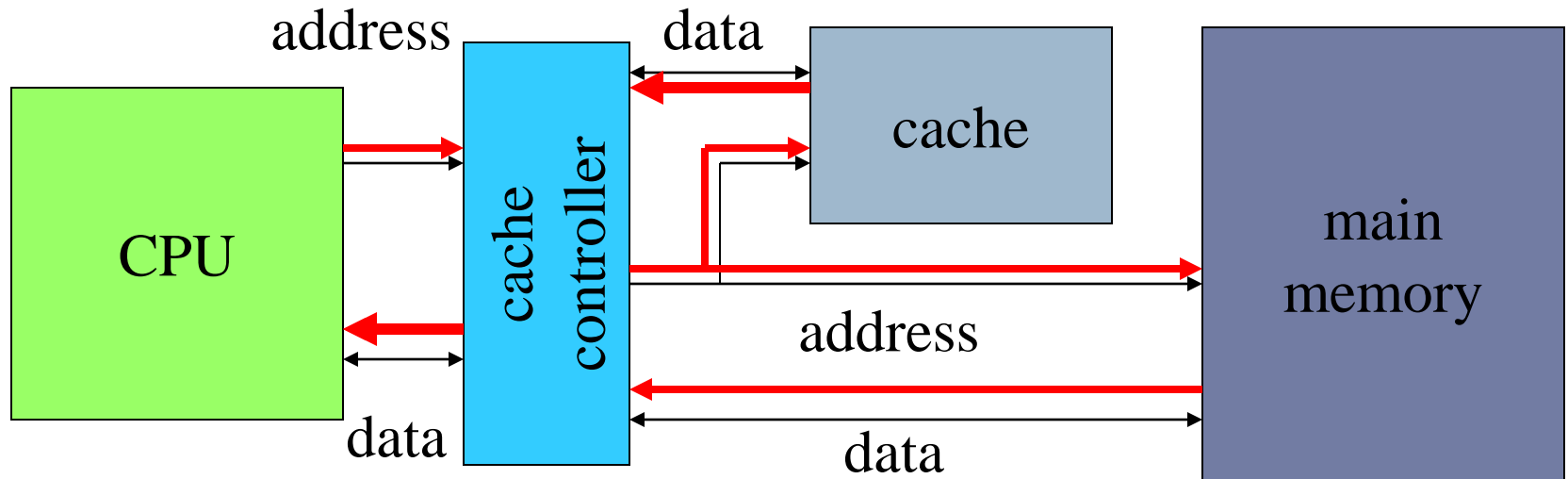


CPUs – Chapter 3.5

- ▶ Caches.
- ▶ Memory management.



Caches and CPUs



Cache operation

- ▶ Many main memory locations are mapped onto one cache entry.
- ▶ May have caches for:
 - ▶ instructions;
 - ▶ data;
 - ▶ data + instructions (**unified**).
- ▶ Memory access time is no longer deterministic.
 - ▶ Depends on “hits” and “misses”



Terms

- ▶ **Cache hit**: required location is in cache.
- ▶ **Cache miss**: required location is not in cache.
- ▶ **Working set**: set of locations used by program in a time interval.



Types of misses

- ▶ **Compulsory (cold)**: location has never been accessed.
- ▶ **Capacity**: working set is too large.
- ▶ **Conflict**: multiple locations in working set map to same cache entry – fighting for the same cache location
- ▶ **Cache miss penalty**: added time due to a cache miss.



Cache performance benefits

- ▶ Keep frequently-accessed locations in fast cache.
- ▶ Cache retrieves more than one word at a time.
 - ▶ Sequential accesses are faster after first access.

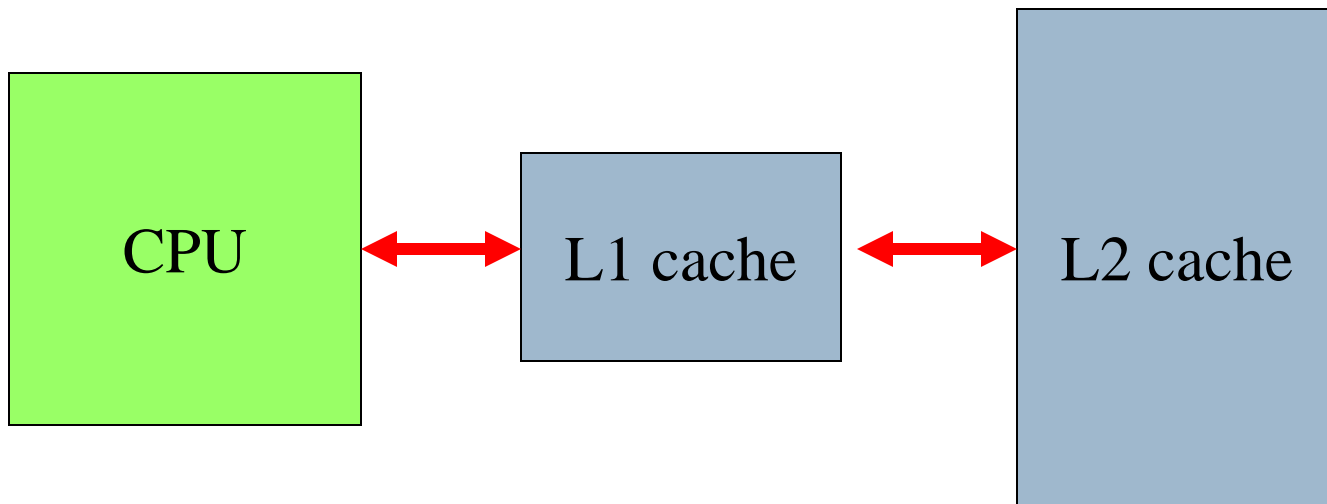


Memory system performance

- ▶ h = cache hit rate.
- ▶ t_{cache} = cache access time, t_{main} = main memory access time.
- ▶ Average memory access time:
 - ▶ $t_{\text{av}} = ht_{\text{cache}} + (1-h)(t_{\text{cache}} + t_{\text{main}})$:look-through
 - ▶ $t_{\text{av}} = ht_{\text{cache}} + (1-h)t_{\text{main}}$:look-aside



Multiple levels of cache



Multi-level cache access time

- ▶ h_1 = cache hit rate.
- ▶ h_2 = rate for miss on L1, hit on L2.
- ▶ Average memory access time:
 - ▶ $t_{av} = h_1 t_{L1} + (h_2 - h_1) t_{L2} + (1 - h_2 - h_1) t_{main}$



Replacement policies

- ▶ **Replacement policy**: strategy for choosing which cache entry to throw out to make room for a new memory location.
- ▶ Two popular strategies:
 - ▶ Random.
 - ▶ Least-recently used (LRU).



Cache organizations

- ▶ **Fully-associative**: any memory location can be stored anywhere in the cache (almost never implemented).
- ▶ **Direct-mapped**: each memory location maps onto exactly one cache entry.
- ▶ **N-way set-associative**: each memory location can go into one of n sets.



Write operations

- ▶ **Write-through**: immediately copy write to main memory.
- ▶ **Write-back**: write to main memory only when location is removed from cache.



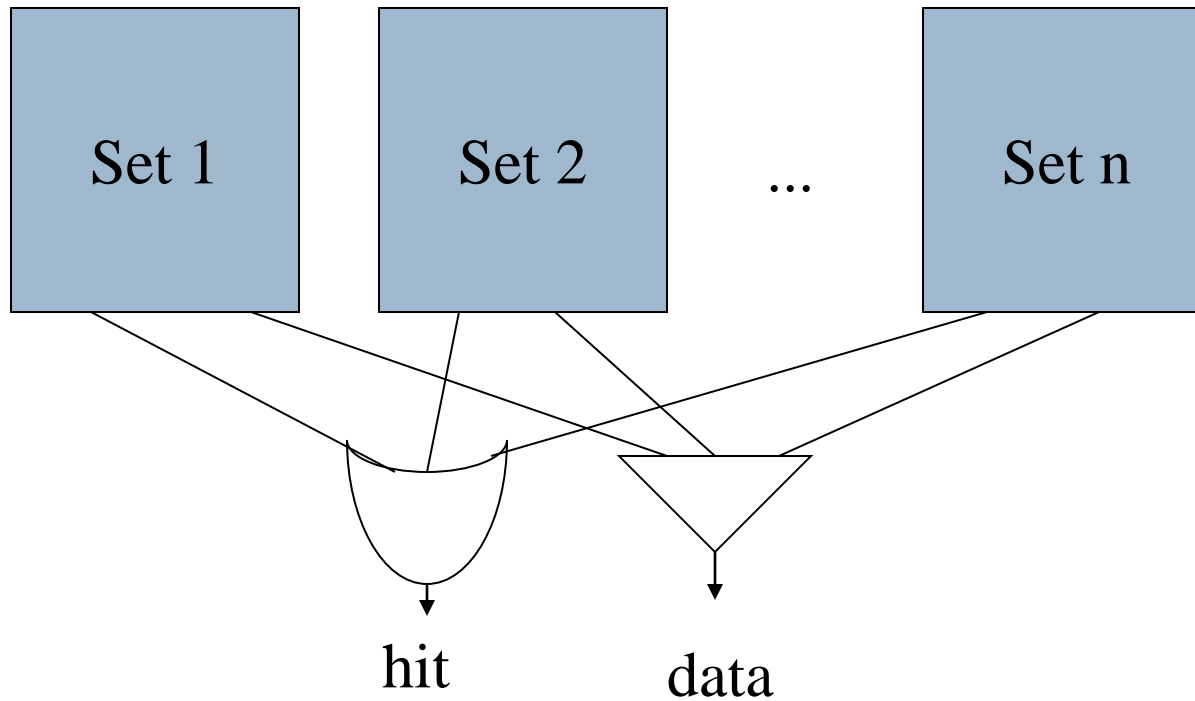
Direct-mapped cache locations

- ▶ Many locations map onto the same cache block.
- ▶ Conflict misses are easy to generate:
 - ▶ Array `a[]` uses locations 0, 1, 2, ...
 - ▶ Array `b[]` uses loc's 0x400, 0x401, 0x402, ...
 - ▶ Operation `a[i] + b[i]` generates conflict misses.



Set-associative cache

- ▶ A set of direct-mapped caches:



Example: direct-mapped vs. set-associative

address

000

001

010

011

100

101

110

111

data

0101

1111

0000

0110

1000

0001

1010

0100



Direct-mapped cache behavior

▶ After 001 access:

block	tag	data
00	-	-
01	0	1111
10	-	-
11	-	-

▶ After 010 access:

block	tag	data
00	-	-
01	0	1111
10	0	0000
11	-	-



Direct-mapped cache behavior, cont'd.

▶ After 011 access:

block	tag	data
00	-	-
01	0	1111
10	0	0000
11	0	0110

▶ After 100 access:

block	tag	data
00	1	1000
01	0	1111
10	0	0000
11	0	0110



Direct-mapped cache behavior, cont'd.

▶ After 101 access:

block	tag	data
00	1	1000
01	1	0001
10	0	0000
11	0	0110

▶ After 111 access:

block	tag	data
00	1	1000
01	1	0001
10	0	0000
11	1	0100



2-way set-associative cache behavior

- ▶ Final state of cache (twice as big as direct-mapped):

set	blk 0 tag	blk 0 data	blk 1 tag	blk 1 data
001		1000	-	-
010		1111		0001
100		0000	-	-
110		0110		0100



2-way set-associative cache behavior

- ▶ Final state of cache (same size as direct-mapped):

set	blk 0 tag	blk 0 data	blk 1 tag	blk 1 data
0	01	0000	10	1000
1	10	0111	11	0100



Example caches

- ▶ **StrongARM:**

- ▶ 16 Kbyte, 32-way, 32-byte block instruction cache.
- ▶ 16 Kbyte, 32-way, 32-byte block data cache (write-back).

- ▶ **C55x:**

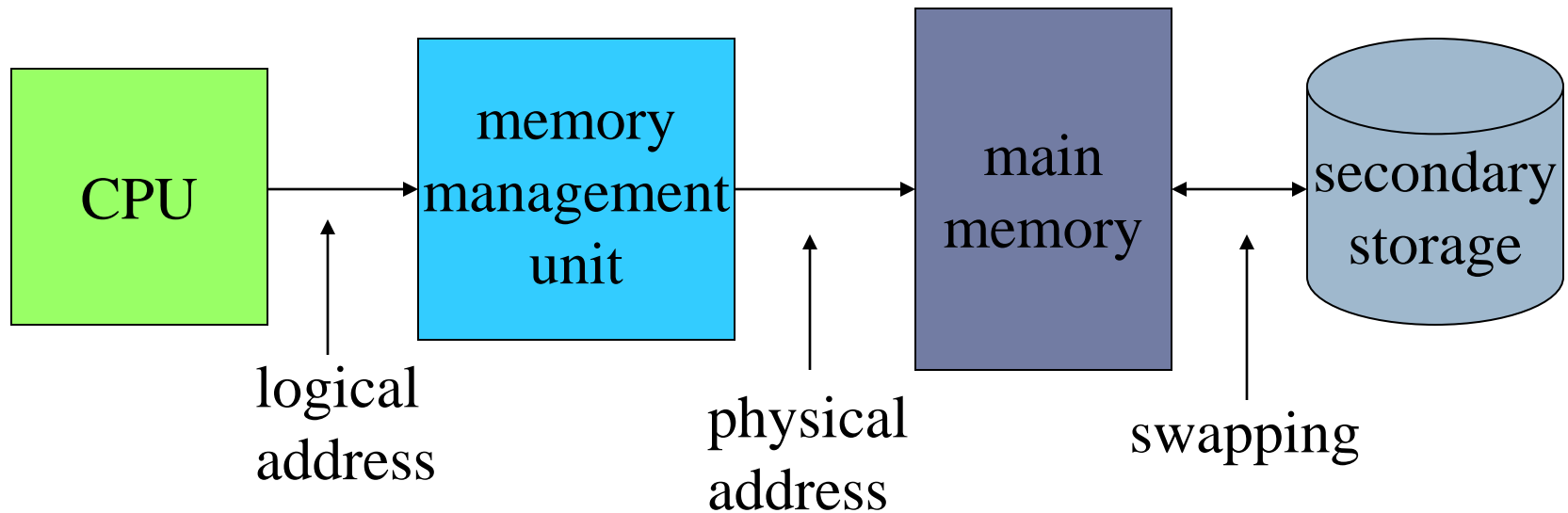
- ▶ Various models have 16KB, 24KB cache.
- ▶ Can be used as scratch pad memory.

Scratch pad memories

- ▶ **Alternative to cache:**
 - ▶ Software determines what is stored in scratch pad.
- ▶ **Provides predictable behavior at the cost of software control.**
- ▶ **C55x cache can be configured as scratch pad.**

Memory management units (3.5.2)

- ▶ Memory management unit (MMU) translates addresses:



Memory management tasks

- ▶ Allows programs to move in physical memory during execution.
- ▶ Allows **virtual memory**:
 - ▶ memory images kept in secondary storage;
 - ▶ images returned to main memory on demand during execution.
- ▶ **Page fault**: request for location not resident in memory.

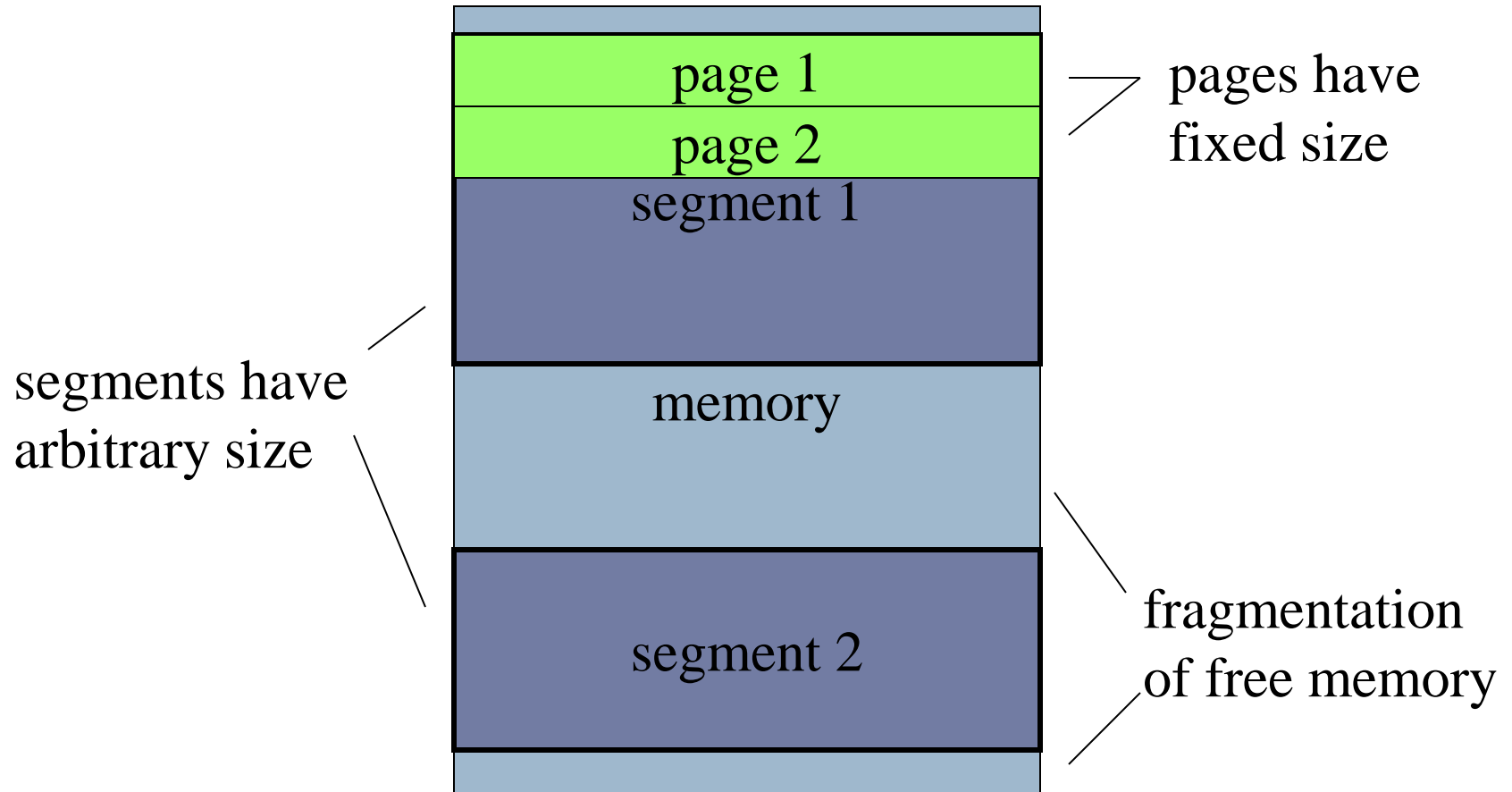


Address translation

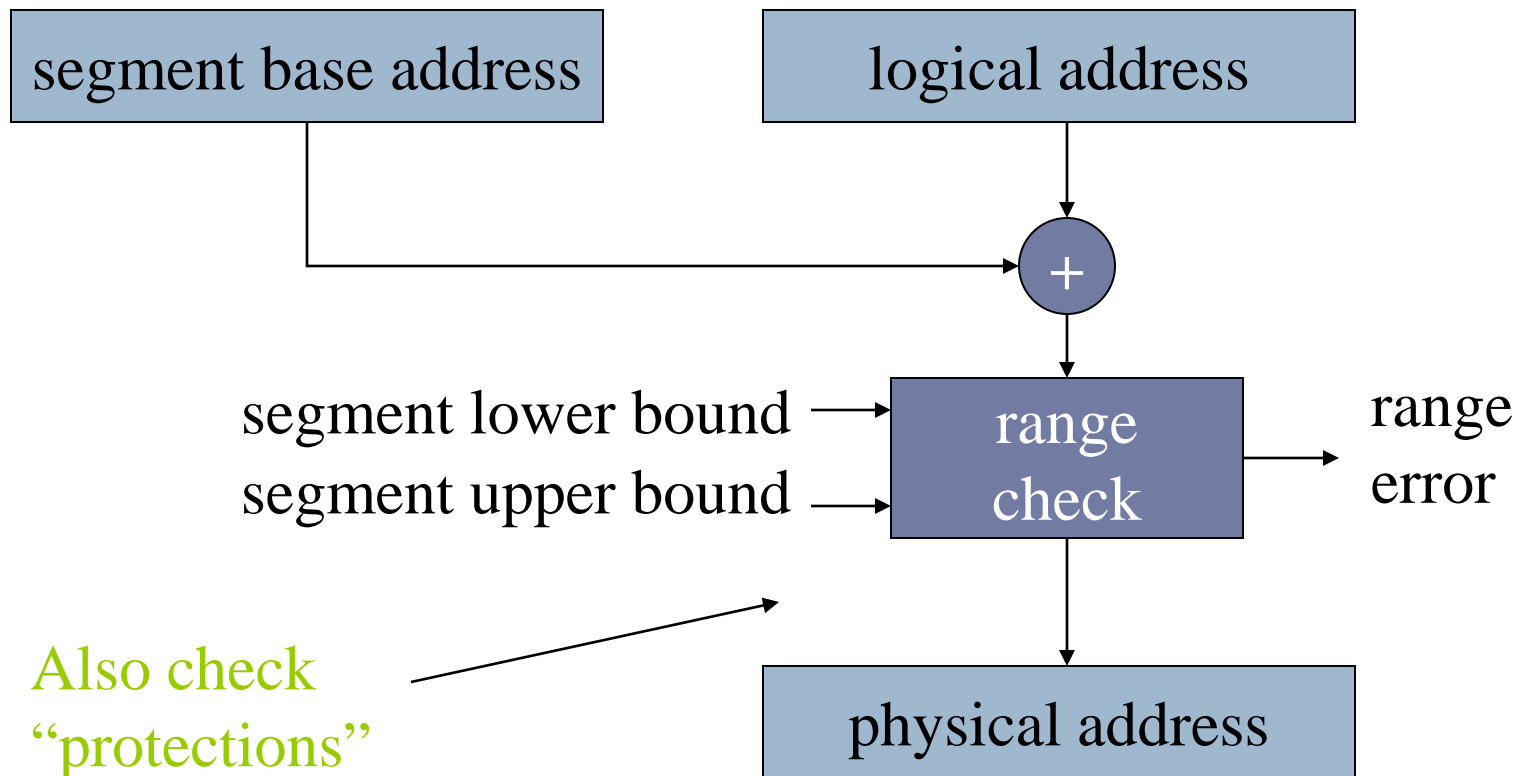
- ▶ Requires some sort of register/table to allow arbitrary mappings of logical to physical addresses.
- ▶ Two basic schemes:
 - ▶ segmented;
 - ▶ paged.
- ▶ Segmentation and paging can be combined (x86, PowerPC).



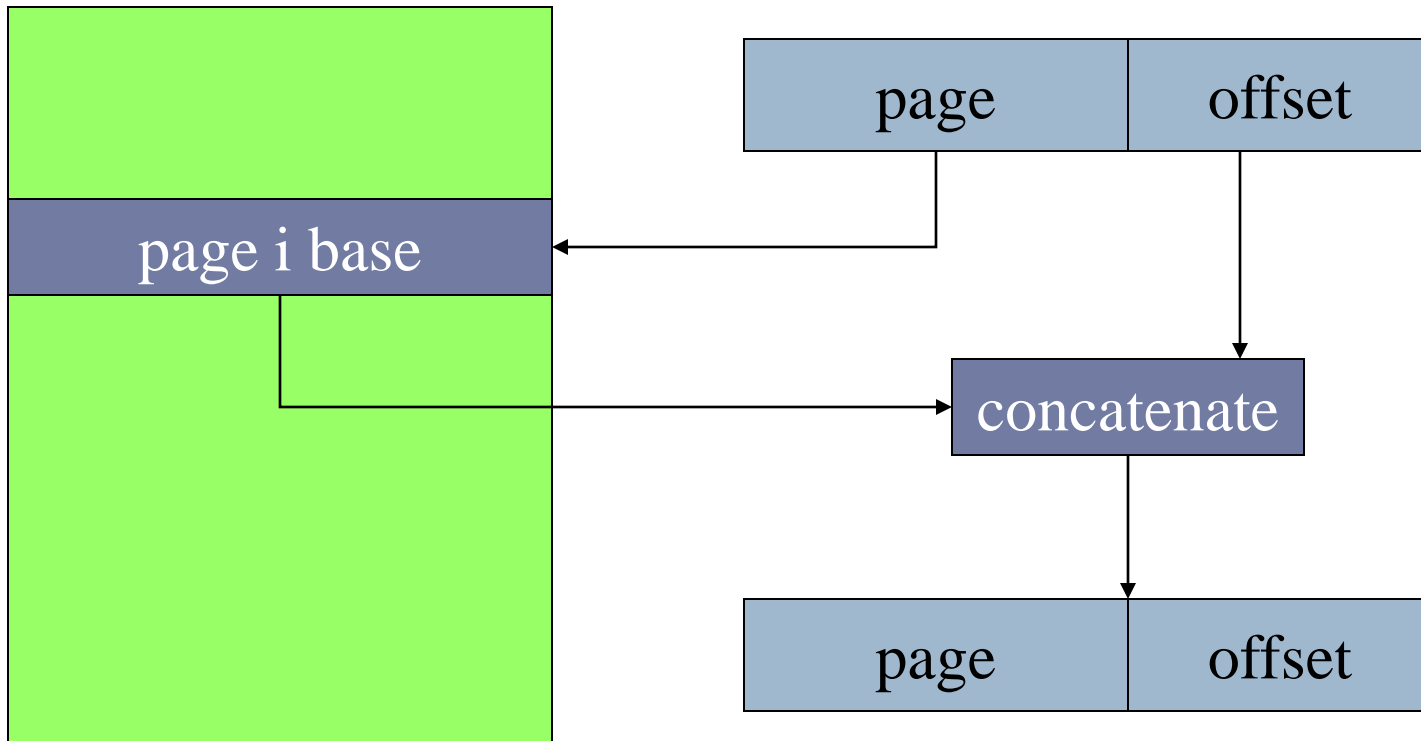
Segments and pages



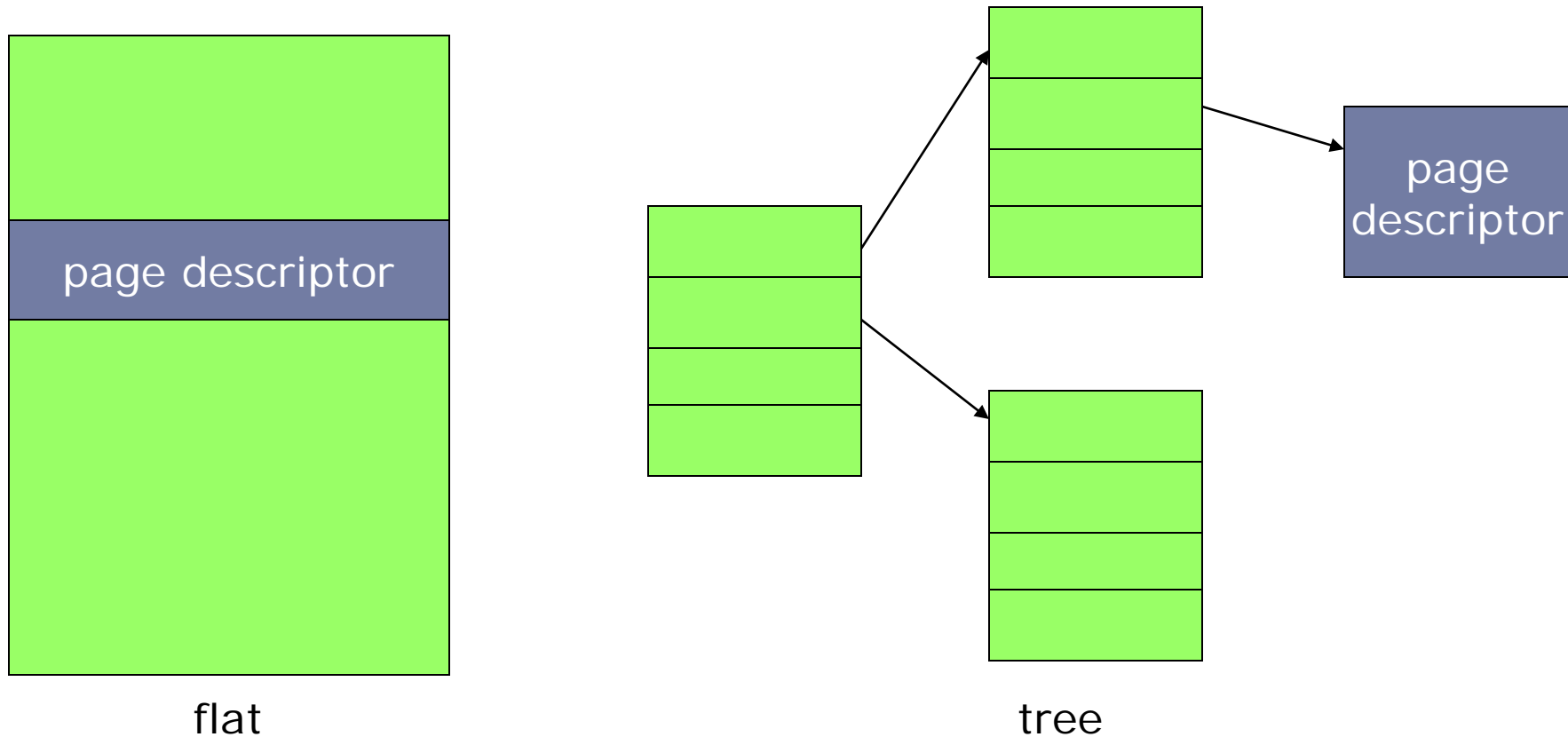
Segment address translation



Page address translation



Page table organizations



Caching address translations

- ▶ Large translation tables require main memory access.
- ▶ **TLB (translation lookaside buffer)**: cache for address translation.
 - ▶ Typically small.



ARM memory management (optional)

- ▶ **Memory region types:**
 - ▶ section: 1 Mbyte block;
 - ▶ large page: 64 kbytes;
 - ▶ small page: 4 kbytes.
- ▶ An address is marked as section-mapped or page-mapped.
- ▶ Two-level translation scheme.



ARM address translation

