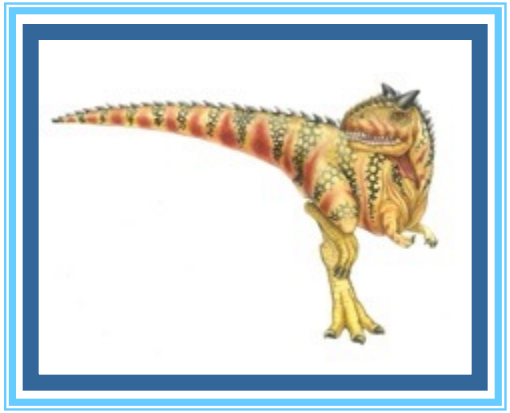
Spring 2022 COMP 3511
Review #7



Coverages

- Memory Management
  - Segmentation
  - Paging
  - Translation look-aside buffer (TLB)

SEGMENTATION

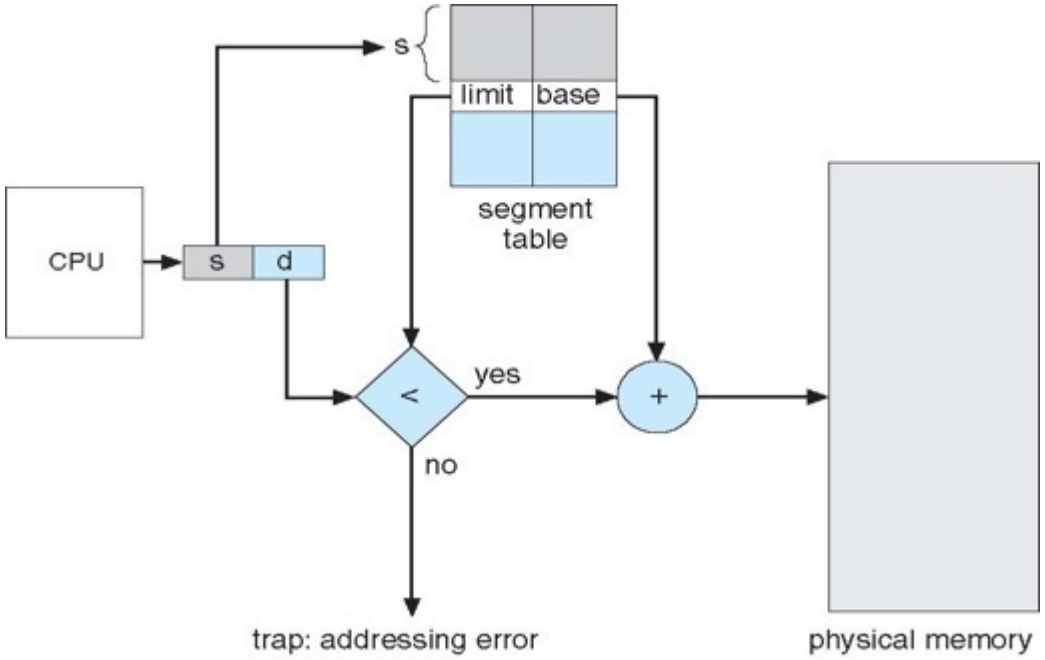
Segmentation Architecture

- The logical address now consists of a two tuple:

  <segment-number, offset>,

- Segment table – maps two-dimensional programmer-defined addresses into one-dimensional physical addresses; each table entry has:
  - base – contains the starting physical address where the segments reside in memory
  - limit – specifies the length of the segment

- Since segments vary in length, memory allocation for a segment is a dynamic storage-allocation problem
- **External fragmentation still exists**, but is much less severe than that in a contiguous allocation



Example of Segmentation

- An example

Segment	Base	Length
0	120	189
1	2155	596
2	525	80
3	1650	64
4	846	378

- What are the physical addresses for the following logical addresses (segment number, offset)?
  - 0, 227
  - 2, 63
  - 3, 45

Example of Segmentation

- An example

Segment	Base	Length
0	120	189
1	2155	596
2	525	80
3	1650	64
4	846	378

- What are the physical addresses for the following logical addresses (segment number, offset)?
  - 0, 227 → Illegal address since 227 > 189
  - 2, 63
  - 3, 45

Example of Segmentation

- An example

Segment	Base	Length
0	120	189
1	2155	596
2	525	80
3	1650	64
4	846	378

- What are the physical addresses for the following logical addresses (segment number, offset)?
  - 0, 227 → Illegal address since 227 > 189
  - 2, 63 → 588 since 525 + 63
  - 3, 45

Example of Segmentation

- An example

Segment	Base	Length
0	120	189
1	2155	596
2	525	80
3	1650	64
4	846	378

- What are the physical addresses for the following logical addresses (segment number, offset)?
  - 0, 227 → Illegal address since 227 > 189
  - 2, 63 → 588 since 525 + 63
  - 3, 45 → 1695 since 1650 + 45

Example of Segmentation

- An example

Segment	Base	Length
0	120	189
1	2155	596
2	525	80
3	1650	64
4	846	378

- What are the logical addresses for the following physical addresses?
  - 247
  - 1450
  - 1043

Example of Segmentation

- An example

Segment	Base	Length
0	120	189
1	2155	596
2	525	80
3	1650	64
4	846	378

- What are the logical addresses for the following physical addresses?
  - 247 → 0, 127
  - 1450
  - 1043

Example of Segmentation

- An example

Segment	Base	Length
0	120	189
1	2155	596
2	525	80
3	1650	64
4	846	378

- What are the logical addresses for the following physical addresses?
  - 247 → 0, 127
  - 1450 → illegal physical address since there is no corresponding logical address
  - 1043

Example of Segmentation

- An example

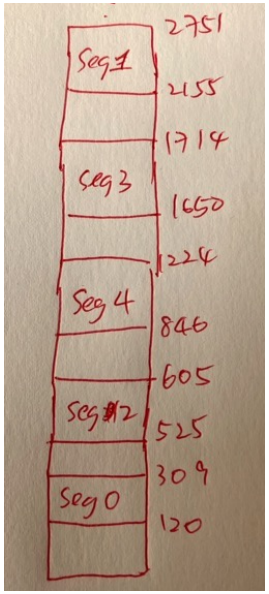
Segment	Base	Length
0	120	189
1	2155	596
2	525	80
3	1650	64
4	846	378

- What are the logical addresses for the following physical addresses?
  - 247 → 0, 127
  - 1450 → illegal physical address since there is no corresponding logical address
  - 1043 → 4, 197

Example of Segmentation

- You can draw the memory layout to help you verify the answers:

Segment	Base	Length
0	120	189
1	2155	596
2	525	80
3	1650	64
4	846	378



- What are the logical addresses for the following physical addresses?
  - 247 → 0, 127
  - 1450 → illegal physical address since there is no corresponding logical address
  - 1043 → 4, 197

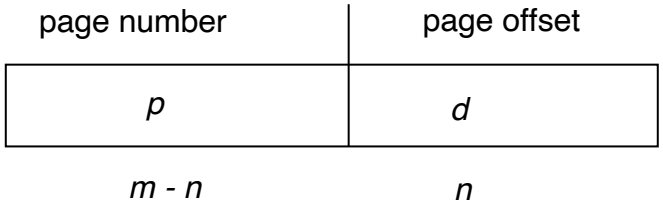
Summary - Segmentation

- Protection is easy in segmentation scheme
  - Protection with segment table
  - code segment would be read-only
  - data and stack would be read-write (stores allowed),
  - shared segment could be read-only or read-write
- The main problem with segmentation
  - It must fit variable-sized chunks into physical memory
    - Dynamic Storage Allocation Problem
  - It may move processes multiple times to fit everything
  - External fragmentation still exists

PAGING

Paging Address Translation

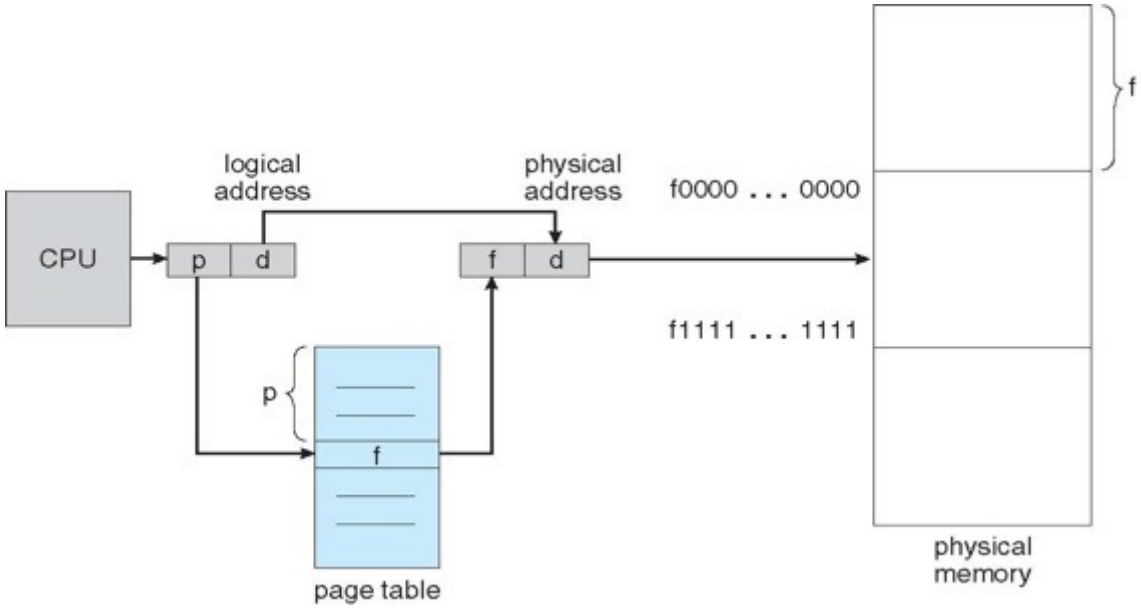
- Divide physical memory into fixed-sized blocks called frames and divide logical memory into blocks of same size called pages
  - Pages has the same size of frames
- Address generated (logical or virtual address) by CPU is divided into:
  - Page number (p) – used as an index into a page table which contains base address (frame number) of each page in physical memory
  - Page offset (d) – combined with base address to define the physical memory address that is sent to the memory unit



- The memory allocation in a paging scheme essentially allocates a frame in the physical memory to a page in a process virtual address space
- For given logical address space 2^m and page size 2^n, thus the number of pages this logical address contains is 2^{m-n}

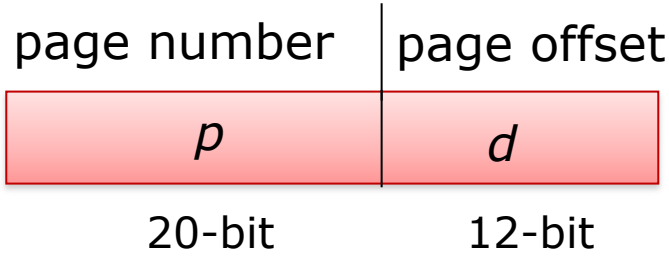
Paging Hardware

- The following outlines the steps taken by the MMU to translate a logical address generated by the CPU to a physical address:
  1. Extract the page number **p** and use it as an index into the page table.
  2. Extract the corresponding frame number **f** from the page table.
  3. Replace the page number **p** in the logical address with the frame number **f**
- As the offset **d** does not change, it is not replaced, and the frame number and offset now comprise the physical address.



Example of Paging Scheme

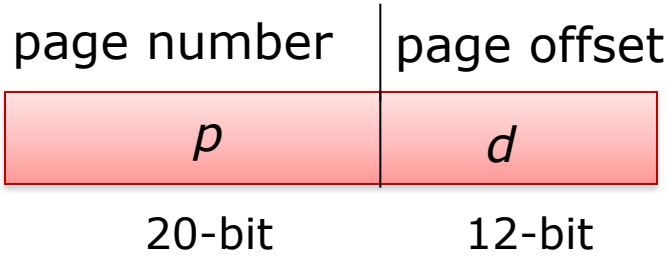
- Suppose logical address of a system is



  - What is the page size in such a system?
  - What is the size of the logical address space?
  - What is the number of pages?

Example of Paging Scheme

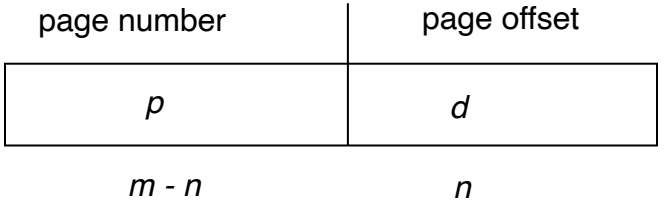
- Suppose logical address of a system is



  - What is the page size in such a system?

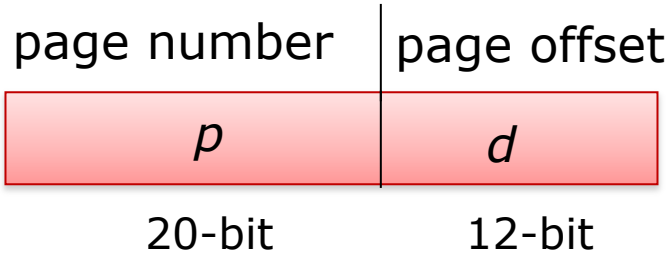
    12 bits are used to offset into a page, the page size is $2^{12} = 4$ KB.

Hints: For given logical address space 2^m and page size 2^n, thus the number of pages this logical address contains is 2^{m-n}



Example of Paging Scheme

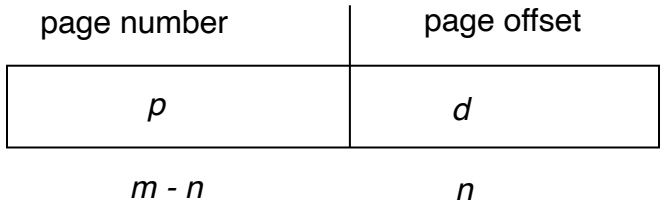
- Suppose logical address of a system is



  - What is the size of the logical address space?

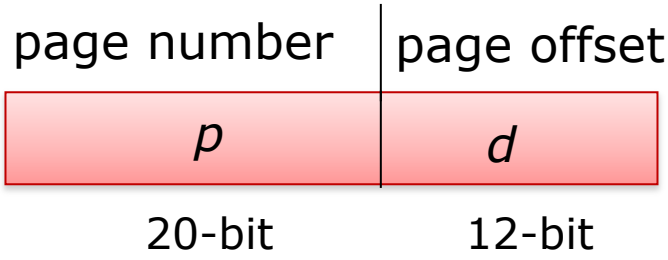
    32 bits virtual address indicates virtual address space is $2^{32} = 4$ GB.

Hints: For given logical address space 2^m and page size 2^n, thus the number of pages this logical address contains is 2^{m-n}



Example of Paging Scheme

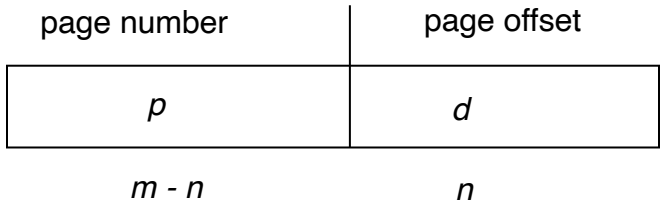
- Suppose logical address of a system is



  - What is the number of pages?

    2^{20} pages

Hints: For given logical address space 2^m and page size 2^n, thus the number of pages this logical address contains is 2^{m-n}.



Example of page address translation

- Suppose that the physical memory size is 4096 bytes. And there is 16 frames in physical memory.
- The page table is listed as follows:

Page No.	Frame No.
0	5
1	2
2	7
3	0

- Please work out physical address of following logical addresses
  - 717
  - 508

Example of page address translation

- Suppose that the physical memory size is 4096 bytes. And there is 16 frames in physical memory.
- The page table is listed as follows:

Page No.	Frame No.
0	5
1	2
2	7
3	0

- Please work out physical address of following logical addresses
  - Each page has 256 bytes (4096/16)

Example of page address translation

- Suppose that the physical memory size is 4096 bytes. And there is 16 frames in physical memory.
- The page table is listed as follows:

Page No.	Frame No.
0	5
1	2
2	7
3	0

- Please work out physical address of following logical addresses
  - 717 → 717 = 256*2+ 205 → 7*256 + 205 = 1997
  - 508

Example of page address translation

- Suppose that the physical memory size is 4096 bytes. And there is 16 frames in physical memory.
- The page table is listed as follows:

Page No.	Frame No.
0	5
1	2
2	7
3	0

- Please work out physical address of following logical addresses
  - 717 → 717 = 256*2 + 205 → 7*256 + 205 = 1997
  - 508 → 508 = 256*1 + 252 → 2*256 + 252 = 764

Example of page address translation

- Suppose that the physical memory size is 4096 bytes. And there is 16 frames in physical memory.
- The page table is listed as follows:

Page No.	Frame No.
0	5
1	2
2	7
3	0

- Please work out logical address of following physical addresses
  - 444
  - 700

Example of page address translation

- Suppose that the physical memory size is 4096 bytes. And there is 16 frames in physical memory.
- The page table is listed as follows:

Page No.	Frame No.
0	5
1	2
2	7
3	0

- Please work out logical address of following physical addresses
  - 444 → 256*1 + 188 → illegal
  - 700

Example of page address translation

- Suppose that the physical memory size is 4096 bytes. And there is 16 frames in physical memory.
- The page table is listed as follows:

Page No.	Frame No.
0	5
1	2
2	7
3	0

- Please work out logical address of following physical addresses
  - 444 → 256*1 + 188 → illegal
  - 700 → 256*2 + 188 → 256*1 + 188 = 444

Two-Level Page Table

- In a 32-bit machine we subdivide the virtual address into 3 parts as follows:

page number		page offset
10	10	12

- First 10 bits of an address is an index into the first level page table.
- The next 10 bits are an index into a second level page table.
- Assume that each page table entry is 64 bits (8 bytes) in size.

Two-Level Page Table

- Q1: What is the advantage of using two-level paging scheme?
  - Two-level paging scheme can divide the large page table into many smaller pieces.
  - Therefore, there is no need to allocate the whole page table contiguously in main memory.

Two-Level Page Table

- Q2: How much space is occupied in memory by the page tables for a process that has 128MB of actual virtual address space allocated?
- Show your work with a detailed explanation.
  - Page size: $2^{12} = 4096B = 4KB$ (1 point)

Two-Level Page Table

- Q2: How much space is occupied in memory by the page tables for a process that has 128MB of actual virtual address space allocated?
- Show your work with a detailed explanation.
  - Page Table Size:
    - For each first-level page table, it has $2^{10} = 1024$ entries.
    - Since each entry is 8B (64 bits) in size, the size of the first-level page table is 8KB.
    - For each second-level page table, its size is also 8KB. (1 point)

Two-Level Page Table

- Page Number:
  - For a process that has 128MB, it has 128MB/4KB=32K pages. (1 point)
- First-level page table:
  - 1 first-level page table needed
- Second-level page table:
  - number of pages / number of entries in a page table
  - 32K/1K = 32 second-level page table. (1 point)
- Therefore, the answer is $(1 + 32) * 8KB = 264KB$ (1 point)

Get Page Size with Linux

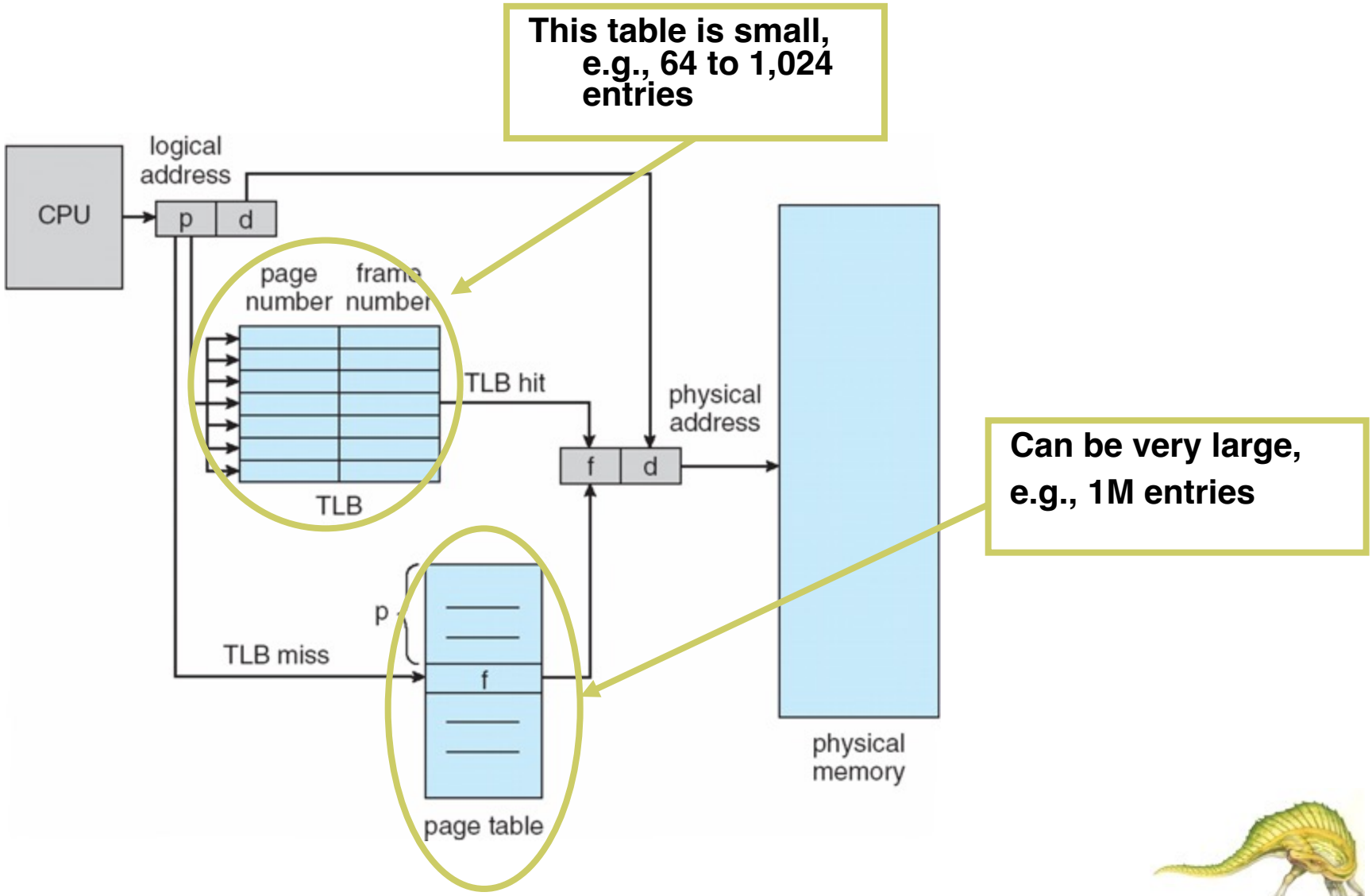
- Use the command `getconf PAGESIZE`
- Use the system call `getpagesize()`

TRANSLATION LOOK-ASIDE BUFFER (TLB)

TLB

- The two memory access problem can be solved by using TLB (translation look-aside buffer) – caching
  - a special, small, fast-lookup hardware cache
  - each entry in the TLB consists of a key (or tag) and a value
  - The page number is presented to the TLB, if found, its frame number is immediately available to access memory
  - fast but expensive – done in hardware
  - Typical size of a TLB 64 – 1,024 entries

Paging Hardware With TLB



Effective Access Time (EAT): Example

- Suppose that a system has a TLB hit ratio of 85%. It requires 10 nanoseconds to access the TLB, and 100 nanoseconds to access main memory. What is the effective memory access time in nanoseconds for this system?

Answer: If the hit ratio is 85%

EAT = (10 + 100) * 0.85 + // Hit: 1 TLB access + 1 memory access
(10 + 200) * 0.15 // Miss: 1 TLB access + 2 memory accesses

= 93.5 + 31.5 = 125

Note: EAT will also be used in Chapter 10: Virtual Memory to evaluate the performance of demand paging. Their calculations are different. Don't mess them up!