state New or Ready-Suspend to state Ready

12.

12.

* Comb^{n} of static & Dynamic decisions.

We decide statically which sizes of blocks to allow (i.e. 2^n), and we decide dynamically how many of each size to have.

Buddy System

- A reasc ercome

12

12.

* Comb^{n} of static & Dynamic decisions.

We decide statically which sizes of blocks to allow (i.e. 2^{n}), and we decide dynamically how many of each size to have.

Buddy System

- **A reasonable *compromise* to overcome disadvantages of both fixed and variable partitioning schemes**
- **Modified forms are used in Unix and Linux**
- **Memory blocks are available in size of 2^{K} where L <= K <= U and where**
  - 2^{L} = smallest size of free block
  - 2^{U} = largest size of free block
  - (generally, the entire memory available)

→ Kernel Mem. Allocn.

- 2^{L} = smallest size of free block
- 2^{U} = largest size of free block
- (generally, the entire memory available)

Buddy System

- **We start with the entire block of size 2^{U}**
- **When a request of size S is made:**
  - If 2^{U-1} < S <= 2^{U} then allocate the entire block of size 2^{U}
  - Else, split this block into two *buddies*, each of size 2^{U-1}
  - If 2^{U-2} < S <= 2^{U-1} then allocate one of the 2 buddies
  - Otherwise one of the 2 buddies is split again
- **This process is repeated until the smallest block greater or equal to S is generated**
- **Two buddies are merged whenever both of them become free**

Buddy System

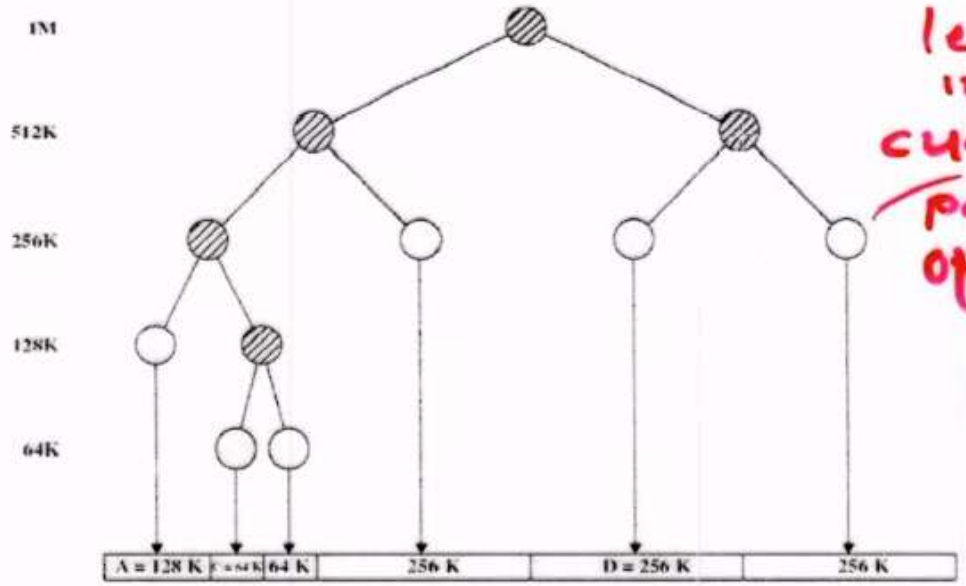
- **The OS maintains several lists of holes**
  - the i-list is the list of holes of size 2^{i}
  - whenever a pair of buddies in the i-list occur, they are removed from that list and merged into a single hole in the (i+1)-list
- **Presented with a request for an allocation of size k such that 2^{i-1} < k <= 2^{i}:**
  - the i-list is first examined
  - if the i-list is empty, the (i+1)-list is then examined...

Example of Buddy System

Example of Buddy System

1 Mbyte block	1 M				
Request 100 K	A = 128 K	128 K	256 K	512 K	
Request 240 K	A = 128 K	128 K	B = 256 K	512 K	
Request 64 K	A = 128 K	C = 64 K	64 K	B = 256 K	512 K
Request 256 K	A = 128 K	C = 64 K	64 K	B = 256 K	D = 256 K	256 K
Release B	A = 128 K	C = 64 K	64 K	256 K	D = 256 K	256 K
Release A	128 K	C = 64 K	64 K	256 K	D = 256 K	256 K
Request 75 K	E = 128 K	C = 64 K	64 K	256 K	D = 256 K	256 K
Release C	E = 128 K	128 K	256 K	D = 256 K	256 K	
Release E	512 K			D = 256 K	256 K	
Release D	1 M				

Tree representation



leaf nodes indicates current partitioning of the Mem.

Figure 7.7 Tree Representation of Buddy System

(after Release B request)

request

Advantages of buddy systems

- **Average memory waste (internal fragmentation) is 25%**
  - on the avg for each program there will be one full power of two partition, plus one half-used
  - plus of course wasted blocks if there aren't any programs waiting small enough for them.
- **Programs are not moved in memory**
  - this simplifies memory management and address translation.

- Application in 11^{el} syst.s as an efficient means of all^{n} & release for 11^{el} prog^{s}.

Recall our discussion about linking/loading...

- **So far, we have only discussed *loading* concepts: how programs are loaded in physical memory.**
- **We have assumed that programs are loaded contiguously in memory.**
- **But programs can consist of several logical parts. These are the segments, or load modules, e.g.:**
  - one or more executable segments
  - one or more data segments
  - stack segment
- **Idea: make allocation more flexible by allocating segments independently.**

Simple Segmentation

(Vs Dynamic Part 149)

- Each program is subdivided into blocks of non-equal size called segments
- these are program modules visible to programmer
- When a process gets loaded into main memory, its different segments can be located anywhere
- The methods for allocating memory to segments are those we have seen so far: just replace *program* by *segment*
- However in this chapter we consider only the case where memory is allocated to segments by using *dynamic partitioning*.
- We can use the methods already studied: e.g. buddy systems

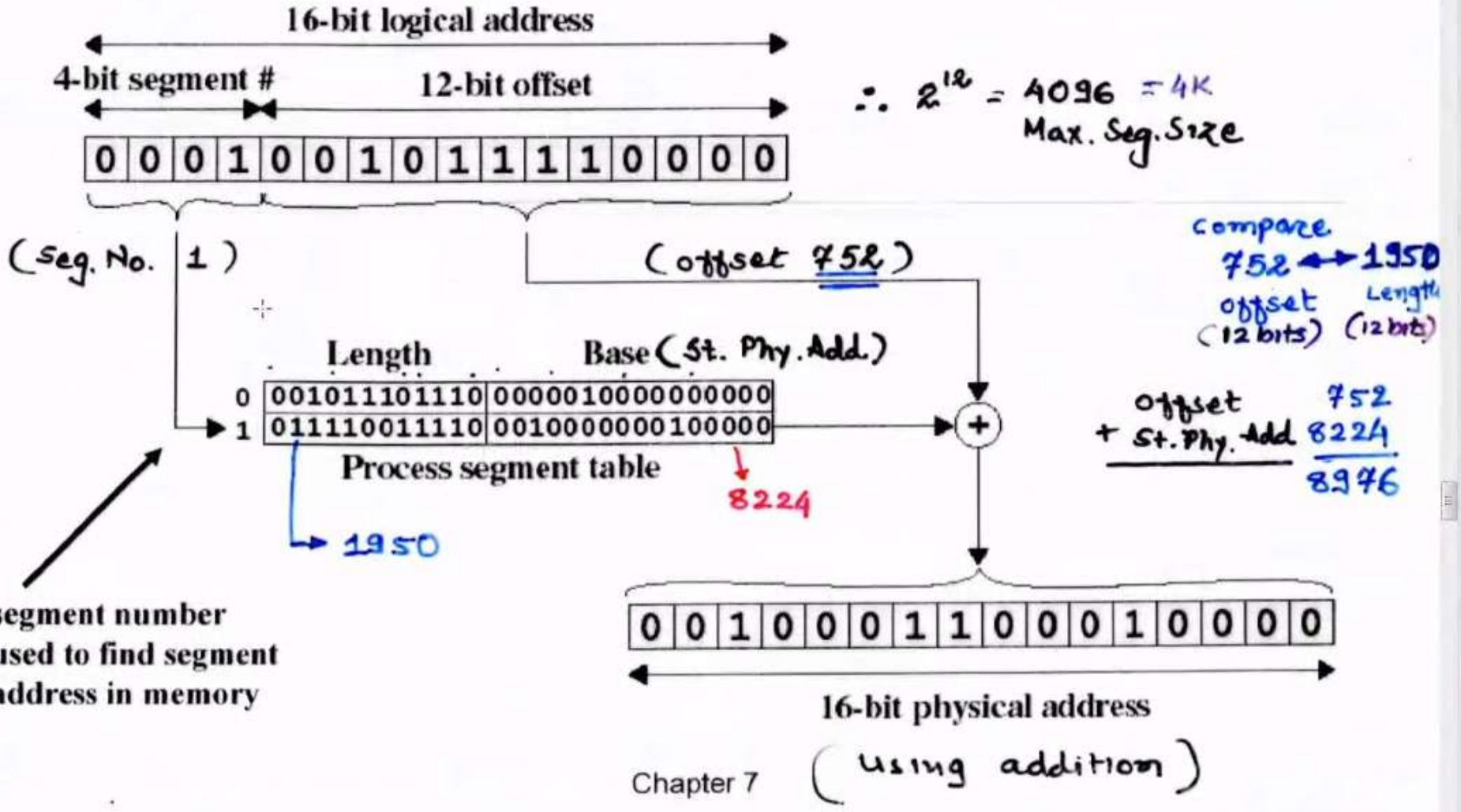
than large ones).

Simple Segmentation

- **The OS maintains a segment table for each program. Each entry contains:**
  - the starting physical addresses of that segment.
  - the length of that segment (for protection)

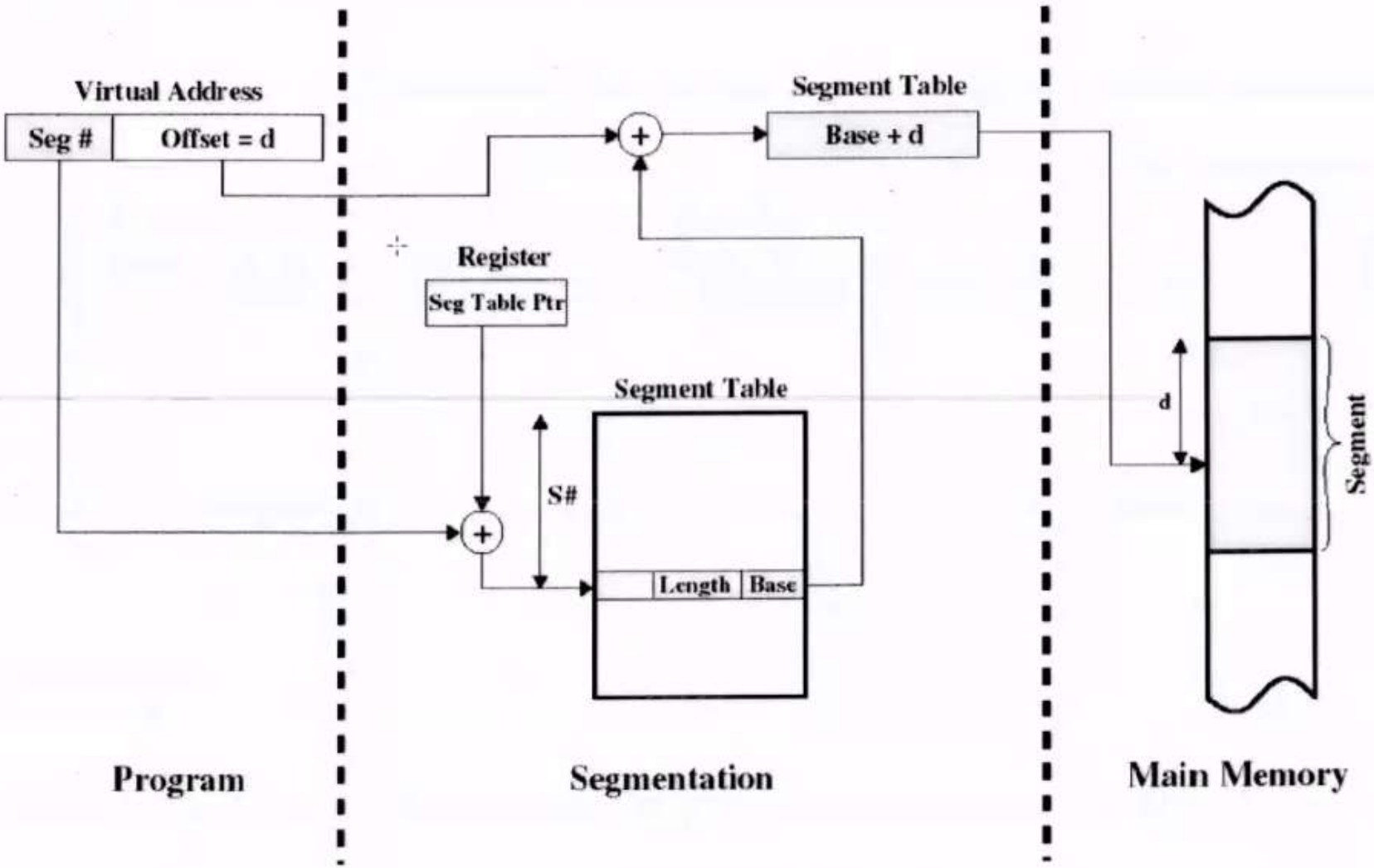
Logical-to-Physical Address translation in segmentation

with dynamic partitioning [redacted]



Address translation in segmentation

(Fig. 3.12 text book)

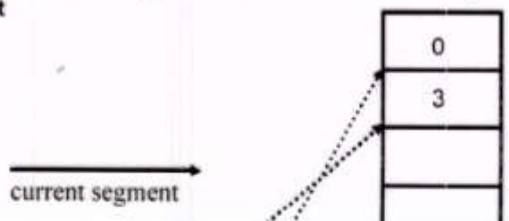


Logical address used in simple segmentation with dynamic partitioning

- **A CPU register holds the starting address of the segment table of the program which the CPU executes.**
- **Presented with a logical address (segment number, offset) = (n,m), the CPU indexes (with n) the segment table to obtain the starting physical address k and the length l of that segment**
- **The physical address is obtained by adding m to k**
  - the hardware also compares the offset m with the length l of that segment to determine if the address is valid

Segmentation mechanisms

- **A table contains the start address of all the segments in a process**
- **Each address in a segment is added to the starting address of that segment**

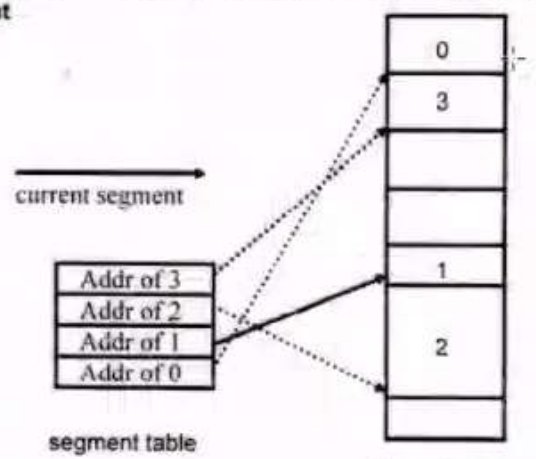


the segment table to obtain the starting physical address k and the length l of that segment

- **The physical address is obtained by adding m to k**
  - the hardware also compares the offset m with the length l of that segment to determine if the address is valid

Segmentation mechanisms

- **A table contains the start address of all the segments in a process**
- **Each address in a segment is added to the starting address of that segment**



Evaluation of Simple Segmentation

- **Advantage: memory allocation unit is a logically independent portion of program**
  - Segments can be loaded individually on demand (*dynamic linking*).
- **Disadvantage: the problems of dynamic partitioning**
  - external fragmentation
  - compaction
- **The next step is to try and simplify mechanisms by using *equally sized* memory allocation units.**

Simple Paging (Vs Fixed Partitioning)

- **Main memory is partitioned into equal fixed-sized chunks called *frames***
- **Each program is also ideally divided into chunks of the same size called *pages***
- **The pages of a program can thus be**