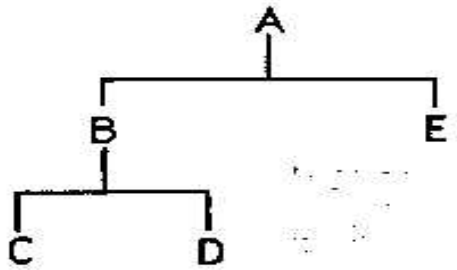
II. Overlays

Programs can be sectioned into modules

Not all modules need to be in main memory at the same time



Programmer specifies which modules can overlay each other

Linker inserts commands to invoke the loader when the modules are referenced

The "parent" must stay in memory

Overlays Continued

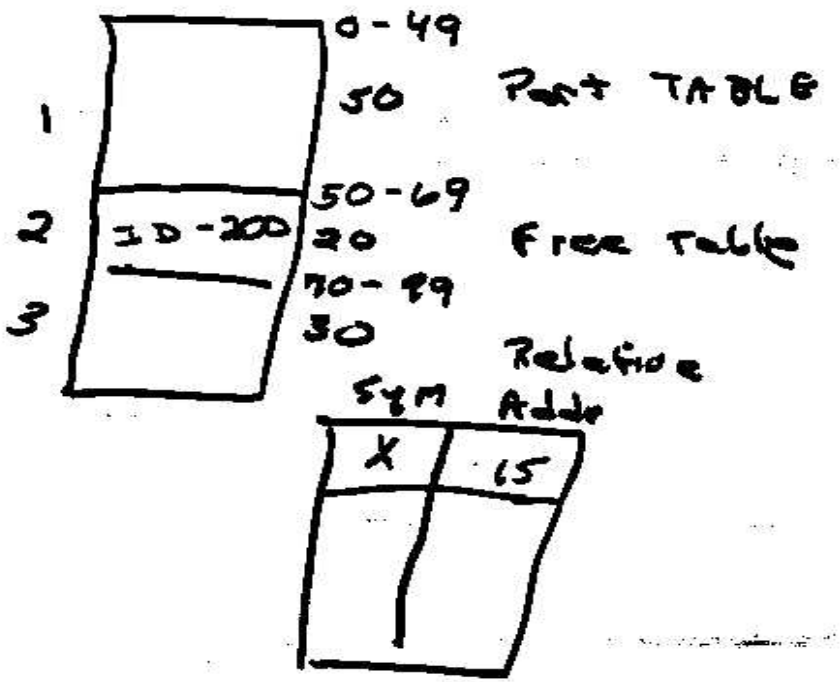
Advantages:

- Reduced memory usage

Disadvantages:

- Overlap map must be specified by programmer
- Programmer must know memory requirements
- Overlapped modules must be completely disjoint

STATIC Partitions
(fixed)



Base + RA → AA

Base Reg - 50
Bounds Reg - 20

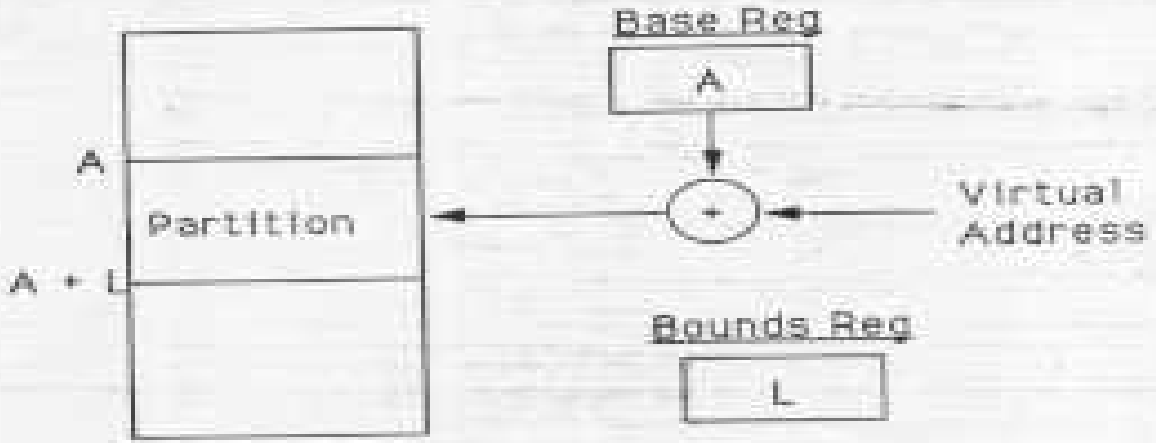
Multiprogramming Protection

Must ensure that a process does not access memory space dedicated to another process

Base Register - Holds the address of the beginning of the partition

Bounds (Limit) Register - Holds the length of the partition

Each relative address is compared to the bounds register and if in range it is added to the base register to produce a physical address

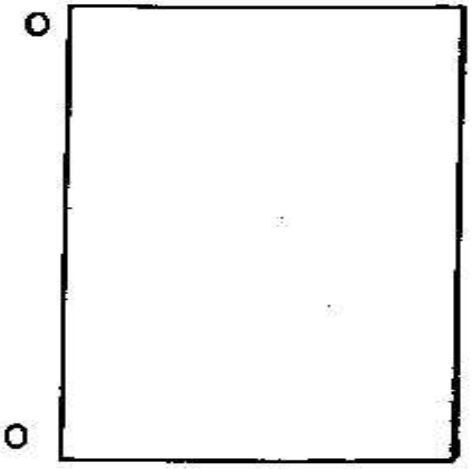


IV. Dynamic Partitions

Consider following scenario (100K memory):

1. Job 1 arrives; size= 22 K
2. Job 2 arrives; size= 24 K
3. Job 3 arrives; size= 30 K
4. Job 4 arrives; size=10 K
5. Job 1 terminates
6. Job 3 terminates
7. Job 5 arrives; size=12K

Let's trace out what happens:



Where should job 5 be put? What are pros/cons of each placement?

Table Sort Illustration

a	FREE - 22 K
b	IN USE - 24 K
c	FREE - 30K
d	IN USE - 10 K
e	FREE - 14 K

First fit:

Start addr	Length
a	22
c	30
e	14

Best fit:

Start addr	Length
e	14
a	22
c	30

Worst fit:

Start addr	Length
c	30
a	22
e	14

Partition Selection Algorithms

Implementation requires a *free block table.*

Sorting table in a particular manner results in a specific selection algorithm:

1) First Fit
   Sort by location

2) Best Fit
   Sort by size (ascending)
   [don't break up big blocks]

3) Worst Fit
   Sort by size (descending)
   [break up big blocks]

Pros/Cons of Dynamic Partitions

Advantages:

Efficient memory usage

Disadvantages:

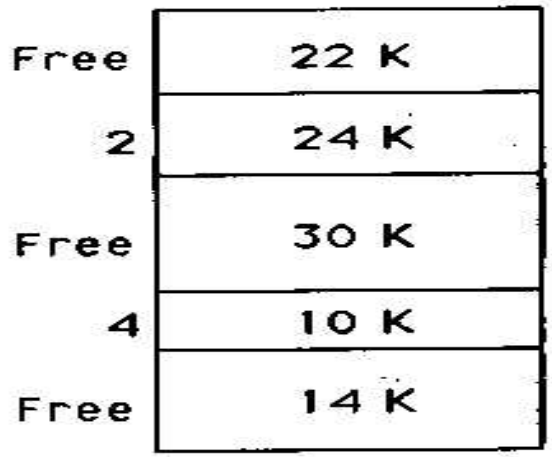
Partition Management

Compaction *or* external fragmentation

What if we cannot find a big enough hole for an arriving job?

Suppose a 35K job arrives?

Suppose a 90K job arrives?



Compaction

Shuffle jobs to create larger contiguous free memory

Do the values of pointers need changing?

Consider the arrival of a 40 K job: