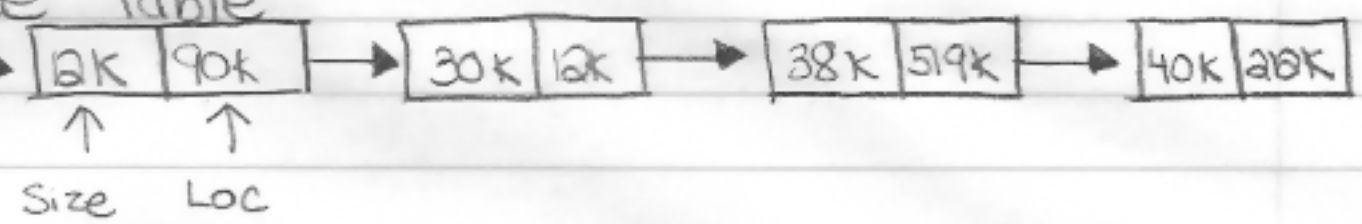
CS 3725 April 3/12

Recap

First-Fit Algorithm : The free table is kept sorted by length.

Best-Fit Algorithm : The free table is kept sorted by size.

eg. Free Table



i) Allocation It tries to find a free partition which fits the job's request as "tightly as possible" (best fit)

Adv: ① In best fit, if exact fit is possible, then that free partition is used.
⇒ Therefore allocation is NOT sensitive to the order of Job Request.

② The search for best fit is logarithmic (binary search) since the free table is kept sorted by size.

ii) Deallocation Here a neighboring free partition is sought.

Disadv: ① It may take a linear search in the worst case.

② Since we find the best possible fit, it leaves very thin free partitions which are big enough to satisfy a job's request.

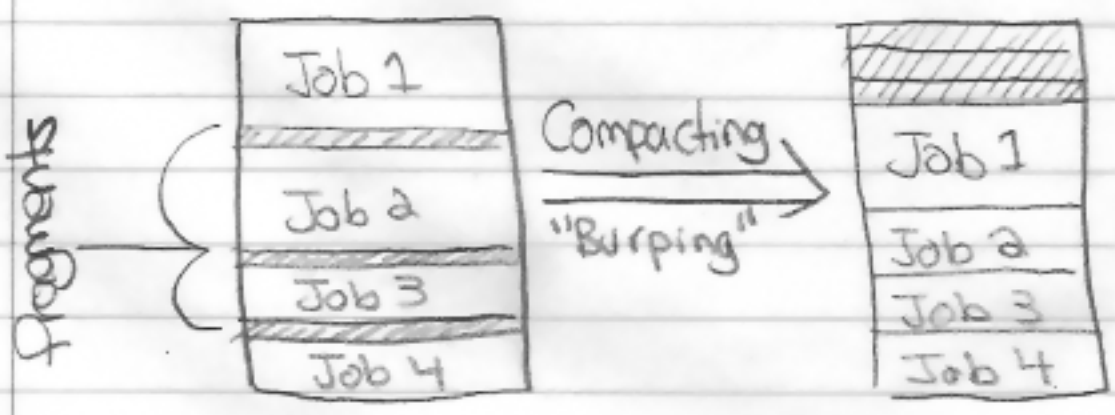
Best Fit + First Fit Problems

① Fragmentation
② If a job has to be swapped out temporarily, it has to be swapped in exact partition.

Next 1) Relocatable Dynamic Partition
2) Paging

Fragmentation

- Due to the allocation schemes in dynamic partitioning
  - A large number of very small free spaces develop throughout the memory.
  - Scattered and isolated, hence wasteful.
  - Lesser degree multiprogrammible



Two differences due to <u>early binding</u>
① It is difficult to determine which datatypes are location sensitive.
② Even if it is possible to determine location sensitive datatypes, the content of the Jobs space will have to be modified.

<u>Large Overhead.</u>
Two Strategies:
① Data type tagging (B5500, 6500)
② Relocatable dynamic partition.

<u>Relocatable Dynamic Partitioning Techniqye</u>
1. Binding between "local address → physical address" is done at the <u>time of execution</u>.

2. The image of a job in memory looks the same as in the logical space.

<u>Technique</u> We maintain two priviledge register:
- Base Register (B)
- Bound Register (b)

The <u>base register</u> gives the first address of the partition belonging to a job.

The <u>bound register</u> gives the last address of the partition belonging to a job.

Binding Let a_e be a logical address and a_p be the corresponding physical address.

$a_p = (a_e + B)$ if $(a_e + B) \leq b$
otherwise out-of-bound error.

→ Examples contrasting IFC without relative dynamic partitions and IFC with relative dynamic partitions.

Machine M°: 1 Address Machine
PC - Prog. Counter
B - Base register
b - Bound register

- ADD X Acc ← Acc + M[x]
- STORE X M[x] = Acc
- BRANCH X PC ← X

I) Without Dyn Relocation

```
PC ← PC+1
IR ← M[PC]
OP ← Dc(IR)
Opr_p ← IA(IR)
Case Op:
    ADD:    ACC ← ACC + M[OPR_p]
    STORE:  M[Opr_p] ← ACC
    BRANCH: PC ← OPR_p
End Case
Loop
```

II) With Relative Dynamic Partitions

$PC_p \leftarrow (PC_e + B)^* + 1$
$IR \leftarrow M[PC_p]$
$OP \leftarrow DC(IR)$
$OPR_e \leftarrow IA(IR)$
Case OP of:
ADD: $ACC \leftarrow ACC + M[OPR_e + B]^*$
STORE: $M[OPR_e + B]^* \leftarrow ACC$
BRANCH:
END
LOOP

*We are not including in the script the out-of-bound test

Hardware Level Implementation