

**WIA2004 OPERATING SYSTEM**

**LAB 1 PROJECT REPORT**

**BEST-FIT**

**GROUP MEMBERS**

| **No.** | **Member Name** | **Matric Number** |
| --- | --- | --- |
| **1.** | **MUHAMMAD IZZAT HAKEEM BIN ROSDI** | **17204856** |
| **2.** | **WAN MUHAMMAD AZIB BIN WAN ABDUL RAHIM** | **17207678** |
| **3.** | **MUHAMMAD AMIRUL AMIN BIN ARMAN** | **17202969** |
| **4.** | **JONE WAN GAN** | **S2031153** |
| **5.** | **LAWRENCE LEROY CHIENG TZE YAO** | **S2018935** |

**OBJECTIVE**

\*Write a program to simulate the following contiguous memory allocation techniques

a) Best-fit

**DESCRIPTION**

One of the simplest methods for memory allocation is to divide memory into several fixed-sized partitions. Each partition may contain exactly one process. In this multiple-partition method, when a partition is free, a process is selected from the input queue and is loaded into the free partition. When the process terminates, the partition becomes available for another process. The operating system keeps a table indicating which parts of memory are available and which are occupied. Finally, when a process arrives and needs memory, a memory section large enough for this process is provided. When it is time to load or swap a process into main memory, and if there is more than one free block of memory of sufficient size, then the operating system must decide which free block to allocate. Best-fit strategy chooses the block that is closest in size to the request. First-fit chooses the first available block that is large enough.

**Best-Fit Memory Allocation**

This method keeps the free/busy list in order by size – smallest to largest. In this method, the operating system first searches the whole of the memory according to the size of the given job and allocates it to the closest-fitting free partition in the memory, making it able to use memory efficiently. Here the jobs are in the order from smallest job to largest job.

**Advantages of Best-Fit Allocation :**

Memory Efficient. The operating system allocates the minimum possible space in the memory, making memory management very efficient. To save memory from getting wasted, it is the best method.

**Disadvantages of Best-Fit Allocation :**

It is a Slow Process. Checking the whole memory for each job makes the working of the operating system very slow. It takes a lot of time to complete the work.

**CODE**

#*BEST-FIT*

# *Function to allocate memory to blocks*

# *as per Best fit algorithm*

# *m number of available blocks*

# *n number of processes*

def bestFit(blockS, m, processS, n):

# *Stores block id of the block*

# *allocated to a process*

allocation = [-1] \* n

# *pick each process and find suitable*

# *blocks according to its size ad*

# *assign to it*

*for* i *in* range(n):

# *Find the best fit block for*

# *current process*

bestIdx = -1

*for* j *in* range(m):

*if* blockS[j] >= processS[i]:

*if* bestIdx == -1:

bestIdx = j

*elif* blockS[bestIdx] > blockS[j]:

bestIdx = j

# *If we could find a block for*

# *current process*

*if* bestIdx != -1:

# *allocate block j to p[i] process*

allocation[i] = bestIdx

# *Reduce available memory in this block.*

blockS[bestIdx] -= processS[i]

print("Process No. Process Size Block no.")

*for* i *in* range(n):

print(i + 1, " ", processS[i],

end=" ")

*if* allocation[i] != -1:

print(allocation[i] + 1)

*else*:

print("Not Allocated")

# *Driver code*

*if* \_\_name\_\_ == '\_\_main\_\_':

blockS = list(int(num) *for* num *in* input(

"Please enter block size: ").strip().split())

processS = list(int(num) *for* num *in* input(

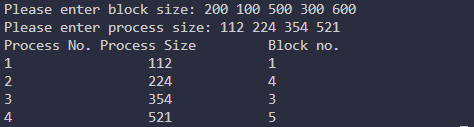
"Please enter process size: ").strip().split())

m = len(blockS)

n = len(processS)

bestFit(blockS, m, processS, n)

**OUTPUT**

****