

**WIA2004 OPERATING SYSTEM**

**LAB 3 PROJECT REPORT**

**CPU**

**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** |

### **QUESTION**

Write a program to simulate the following file allocation strategies.

1. Sequential

### **OBJECTIVE**

Write a program to simulate the following file allocation strategies.

1. Sequential

### **DESCRIPTION**

A file is a collection of data, usually stored on disk. As a logical entity, a file enables to divide data into meaningful groups. As a physical entity, a file should be considered in terms of its organization. The term "file organization" refers to the way in which data is stored in a file and, consequently, the method(s) by which it can be accessed.

### **SEQUENTIAL FILE ALLOCATION**

In this file organization, the records of the file are stored one after another both physically and logically. That is, record with sequence number 16 is located just after the 15th record. A record of a sequential file can only be accessed by reading all the previous records.

### **SFA**

* + SFA is using contiguous file allocation method.
  + Both sequential and direct accesses are supported by this method.
  + This method is extremely fast since the number of seeks are minimal due to the contiguous allocation of file blocks
  + This method suffers from both internal and external fragmentation which makes it inefficient in terms of memory utilization.
  + Increasing file size is difficult because it depends on the availability of contiguous memory at a particular instance.

### **CODE**

def main():  
 # generate files array  
 files = []  
 for x in range(0, 20):  
 files.append(0)  
 fileAllocation(files)  
  
def fileAllocation(files):  
 # inputting file from the user  
 startBlock = int(input("Enter the starting block: "))  
 length = int(input("Enter the length of file: "))  
 # Check whether the file can be allocated to the disk  
 flag = 0  
 for i in range(startBlock, startBlock+length):  
 if files[startBlock-1] == 0:  
 flag += 1  
 # if flag == length of file, proceed  
 if flag == length:  
 for j in range(startBlock, startBlock+length):  
 if files[j - 1] == 0:  
 files[j - 1] = 1  
 print("Position "+str(j)+": "+str(files[j - 1]))  
 if j == startBlock+length-1:  
 print("The file is allocated to the disk")  
 else:  
 print("The file is not allocated to the disk")  
 # Check if the user wants to add more files  
 choice = int(input("Do you want to enter more files? (1 for YES, 0 for NO)"))  
 if choice == 1:  
 fileAllocation(files)  
 else:  
 return  
  
  
# Press the green button in the gutter to run the script.  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

### 

### **OUTPUT**

