K. J. Somaiya College of Engineering, Mumbai-77
(A Constituent College of Somaiya Vidyavihar University)
Department of Computer Engineering

Exp 9 C2 16010121110

TITLE: Implementation of Memory Management Using Address Translation

AIM: To understand process of Address Translation in Memory

Expected Outcome of Experiment:

CO 5. Understand Storage management with allocation, segmentation & virtual memory concepts

Books/ Journals/ Websites referred:

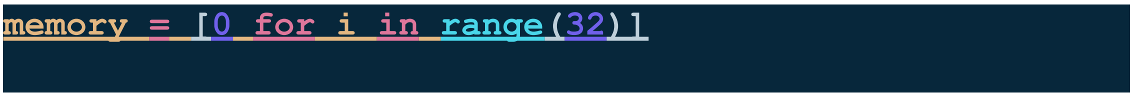
1. **Silberschatz A., Galvin P., Gagne G. "Operating Systems Principles", Willey Eight edition.**
2. **Achyut S. Godbole , Atul Kahate "Operating Systems" McGraw Hill Third Edition.**
3. **William Stallings, "Operating System Internal & Design Principles", Pearson.**
4. **Andrew S. Tanenbaum, "Modern Operating System", Prentice Hall.**

Pre Lab/ Prior Concepts:
Knowledge about the types of memory

Stepwise-Procedure:

1) Implementation of best fit
2) Implementation of next fit
3) Implementation of worst fit

Implementation details

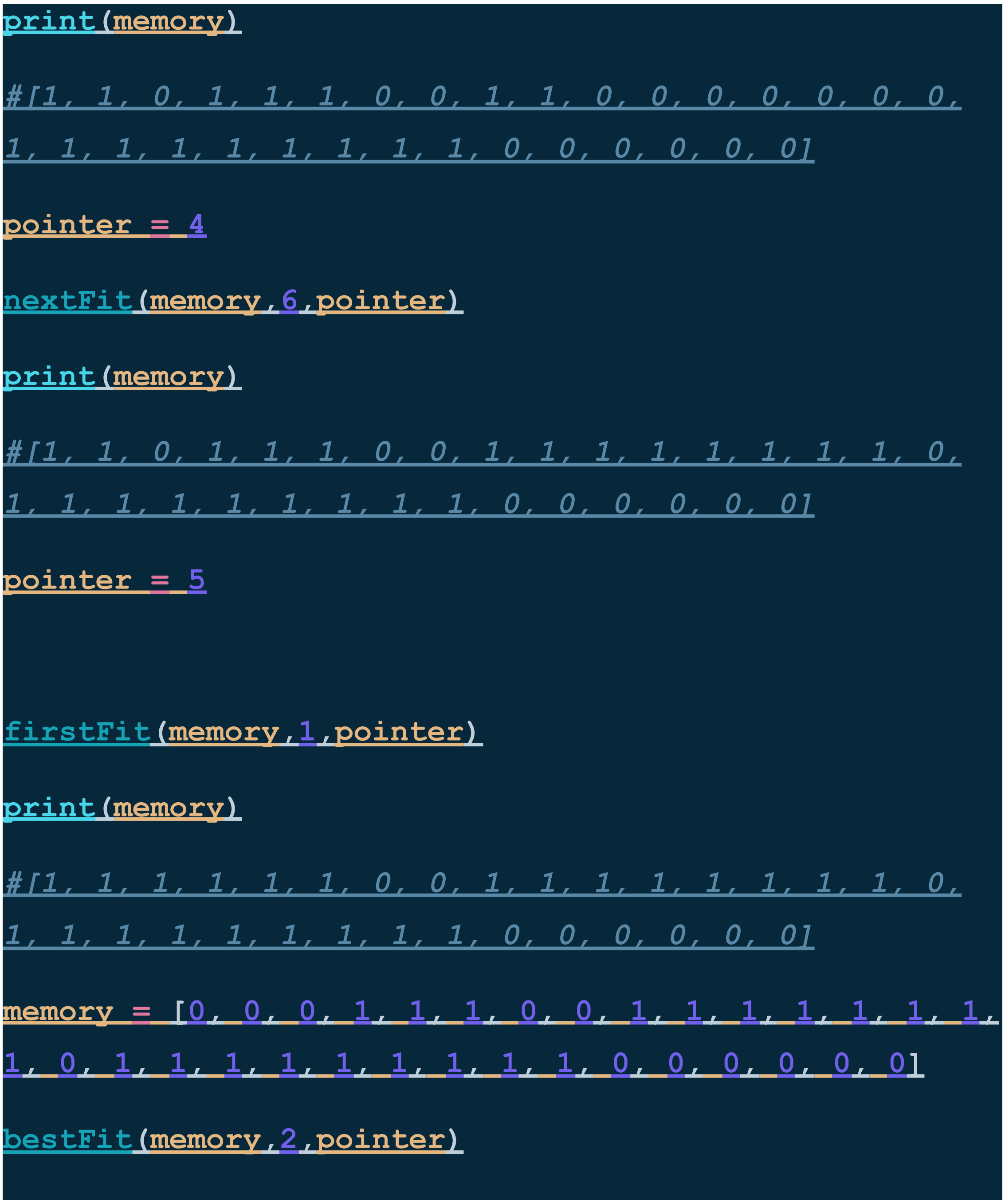


```
memory = [0 for i in range(32)]
```

```
pointer = 0

def nextFit(memory, process_size,pointer):

for ptr in range(pointer, pointer + len(memory)):

ptr = ptr % len(memory)

if(ptr + process_size > len(memory)):

continue

if(sum(memory[ptr : ptr + process_size]) == 0): # is all 0

#allocate

print(ptr)

for i in range(ptr , ptr + process_size):

memory[i] = 1

return (memory,ptr + process_size)


print("memory full")

return -1


def firstFit(memory, process_size,pointer):

nextFit(memory, process_size,0)
```

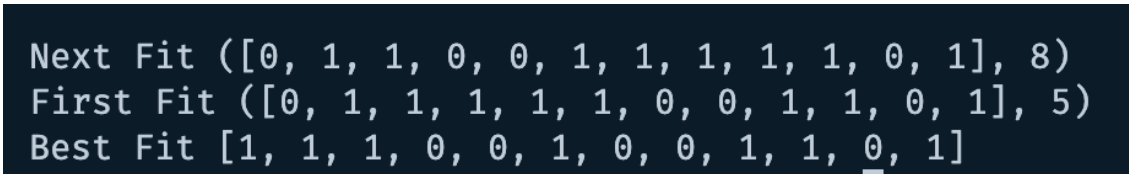
```
def bestFit(memory, process_size,pointer):

memarray = []

memscore = []

ptr = 0

while(True):

mem,p = nextFit(memory.copy(), process_size,ptr)

score = 0

if(mem in memarray):

break # repeat reached



for i in range(p,len(mem)):

if(mem[i]==0):

score +=1

else:

break

try:# for overflow

if(memory[p - process_size -1] ==0): #if the position
does not align at the start of memory chunk
```

```
score = 10000

except:

pass

memarray.append(mem)

memscore.append(score)

ptr+=1

print(memscore)

print(memarray)

print(memscore.index(min(memscore)))

memory = memarray[memscore.index(min(memscore))]

print(memory)

return memarray[memscore.index(min(memscore))]

#randomly fill array

nextFit(memory,2,pointer)

nextFit(memory,3,pointer+3)

nextFit(memory,2,pointer+8)

nextFit(memory,4,pointer+17)

nextFit(memory,5,pointer+13)
```



```
print(memory)

#[1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0,
1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0]

pointer = 4

nextFit(memory,6,pointer)

print(memory)

#[1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0,
1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0]

pointer = 5



firstFit(memory,1,pointer)

print(memory)

#[1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0,
1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0]

memory = [0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1,
1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0]

bestFit(memory,2,pointer)
```

Output: - FF,NF,BF



```
Next Fit ([0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1], 8)
First Fit ([0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1], 5)
Best Fit [1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1]
```

Conclusion: Thus we have implemented the memory allocation algorithms. These algorithms are used for the effective allocation of memory space. This is done to reduce the fragmentation.

Date: 30 oct 2023 **Signature of faculty in-charge**