**EXP NO :10 Virtual Memory: Paging and Segmentation**

**Paging:**

**Aim:**

To design a C program to implement paging

**Algorithm**:

step 1: Start

Step 2: Read the page size for the given memory size.

Step 3: Calculate the number of pages from pagesise and memory size and display it.

Step 4: Form a page table by reading the frame number for every page number.

Step 5: Read the logical address by reading the values of page number and offset value.

Step 6: Check the page number and offset values are less than values of number of pages and page size respectively. If yes, go to step 7, else go to step 8.

step 7: Calculate physical address by (frameno\*pagesize) + offset and display it.

Step 8: Stop.

**Code**:

#include<stdio.h>

void main()

{

int memsize = 15;

int pagesize,noofpage;

int p[100];

int frameno,offset,phyadd,i,choice = 0,pageno;

printf("Your memory size is : %d\n",memsize);

printf("Enter page size: ");

scanf("%d",&pagesize);

noofpage = memsize/pagesize;

printf("The total number of pages is : %d ",noofpage);

for(i=0;i<noofpage;i++)

{

printf("\nEnter the frame of page %d:",i+1);

scanf("%d",&p[i]);

}

do

{

printf("\nLogical address:");

printf("Enter the page number and the offset value\n");

scanf("%d",&pageno);

scanf("%d",&offset);

pageno -= 1;

if(pageno<noofpage && offset < pagesize)

{

phyadd = (p[pageno]\*pagesize)+offset;

printf("\nPhysical address is : %d",phyadd);

printf("\nDo you want to continue?(1/0) ");

scanf("%d",&choice);

}

else

{

printf("Enter the pageno value less than %d and offset value less than %d\n",noofpage,pagesize);

choice = 1;

}

}while(choice == 1);

}

**Output:**

Your Memory size is : 15

Enter page size : 5

The total number of page is : 3

Enter the frame of the page 1: 5

Enter the frame of the page 2: 6

Enter the frame of the page 3: 7

Logical Address:

Enter the page number and the offset value:

1

4

Physical address is :29

**SEGMENTATION**

**Aim:**

To design a c program to implement segmentation in virtual memory.

**Algorithm:**

Step 1: Start.

Step 2: Read the number of segments needed.

Step 3: Read the size and base address for each segments.

Step 4: Enter the data for each parts in a segments according size of a segment.

Step 5: Read the segment number and offset value for recollecting the data from segment.

Step 6: Check the offset and size of the segment of corresponding segment number. If offset is less than size go to step 7. Otherwise goto step 8.

Step 7: Add the offset value with the base address to recollect the appropriate data and display it.

Step 8: Stop.

**Code**:

#include<stdio.h>

int main()

{

int a[10][10],b[100],i,j,n,x,base,size,seg,off;

printf("Enter the segments count\n");

scanf("%d",&n);

for (i=0;i<n;i++)

{

printf("Enter the %d size\n",i+1);

scanf("%d",&size);

a[i][0]=size;

printf("Enter the base address:\n");

scanf("%d",&base);

a[i][1]=base;

for(j=0;j<size;j++)

{

x=0;

scanf("%d",&x);

base ++;

b[base]=x;

}

}

printf("Enter the segment number and the offset value\n"); //To map the segment in the segment table

scanf("%d %d",&seg,&off);

if(off<a[seg][0])

{

int abs=a[seg][1]+off;

printf("The offset is less than %d ",a[seg][0]);

printf("\n %d + %d = %d\n",a[seg][1],off,abs);

printf("The element %d is at %d ",b[abs+1],abs);

}

else

printf("Eroor in locating");

return 0;

}

**Output:**

Enter the segment count = 2

Enter the segment 1 size = 3

Enter the base address = 10

1

2

3

Enter the segment 2 size = 2

Enter the base address = 20

5

6

Enter the segment number and the offset value:

1

1

The offset value is less than 2

20 + 1 = 21

The element 6 is at 21

**Result:**

Thus, the implementation of paging and segmentation in virtual memory is done experimentally and the output is verified.