Assignment - 6

Name - Sahil Chaudhari

Que A[3,2] base address is 1000 and width is 2 calculate the memory address of location A[1,1] using (CMO)

1) $Loc(A[1,1]) = 1000 + 2[3 \times (1-0) + (1-0)]$
$= 1000 + 2[3 \times (1) + (1)]$
$= 1000 + 2[3 + 1]$
$= 1000 + 8$
$= 1008$

2) $Loc[A[2,0]) = 1000 + 2[3 \times (2-0) + (2-0)]$
$= 1000 + 2[3 \times (0) + (2)]$
$= 1000 + 2[0 + 2]$
$= 1000 + 4$
$= 1004$

Que A = [-2 -- 2, 3 -- 7] of element the Starting location is 2000. Each element occupies two memory cell. Calculate the location of A[0,5] using ((MO).

1) $Loc(A=[-1,4]) = 2000 + 2[5 \times (4-3) + (-1-(-2)]$
$= 2000 + 2[5 \times (1) + (1)]$
$= 2000 + 2[5+1]$
$= 2000 + 12$
$= 2012$

2) $Loc((A=[2,6]) = 2000 + 2[5 \times (6-3) + (2-(-2)]$
$= 2000 + 2[5 \times (3) + (4)]$
$= 2000 + 2[15+4]$
$= 2000 + 2[19]$
$= 2000 + 38$
$= 2038$

3) $Loc(A=[0,6]) = 2000 + 2[5 \times (6-3) + (0-(-2$
$= 2000 + 2[5 \times (3) + (2)]$
$= 2000 + 2[15+2]$
$= 2000 + 2[17]$
$= 2000 + 34$
$= 2034$

Multistack write a algorithm for 2 stack push, pop, & traverse function

Step 1 → global declaration stacks
#defind size 10
stack[size];
top1 = -1 , top 2 = stack size.

Step 2 → ~~over~~ flow & push element

```
if (Stack[top1] != Stack[top2])
{ Stack[top1] = insert element
    top 1 ++;
  Stack [top2] = insert element
    top2 --;
    exit;}
else {
    print("stack one or two is overflow")
  }
```

Step 3 → under flow or pop element

```
if ( stack[top1] < 0)
{ print ("stack is underflow")}
else {
Stack[top1] = delete
    stack top = top-1; // assign
}
```

if (Stack [top2] > Stack[max size-1]
{ print ("stack is under flow")}
else {
Stack [top2] = delete;
Stack ⇒ top2 = Top2 + 1; // assign
exit;
}

Step 4 → traverse

if (Top1 == -1 or Top2 > Stack -
[max size - 1]) {
print ("Stack is under flow")
else {
for (i=0; i< Top2; i++) {
print ("Traverse stack(1)" Stack[i])
}
for.
for (i = Top1 + 1; i < Stack of max size; i++)
print ("Traverse stack(2)" Stack[i]);
}
exit;