Project 2

### Design:

My design for my project is fairly simple. Initially the program requests the user to input both n and k. After gathering this information and storing it the program checks if the values are equal to each other, and if so it proceeds to do the factorial of n and outputs that as the final result. If both numbers are not equal the program runs a loop that will go through each (n-k) and multiply it with the previous sum, and then decrement k. After doing this loop the program will then do the final n times the sum of all the previous multiplications. This total will then be displayed to the user.

### Implementation:

The following is my implementation in C

Int N

Int K

Int Total = 1;

printf("Enter n:");

scanf("%d", &n);

printf("Enter k:");

scanf("%d", &k);

if(k==n){

while(n!=0){

total=total\*n;

n=n-1;

}

}

else{

while(k > 1){

k=k-1;

total=total\*(n-k);

}

if(k=1){

total = total\*n;

}

}

printf("Permutation is: %d", total);

return 0;

}

When moving this implementation into assembly code I essentially did the same as the above code but I had to frequently move the values back and forth between memory and the registers in order to complete all the calculations using the amount of registers available.

.586

.MODEL FLAT

EXTRN \_printf:PROC

EXTRN \_scanf:PROC

.STACK 4096 ; reserve 4096-byte stack

.DATA

x DWORD 5 DUP (?) ;used for temporary storage of input

total DWORD ?

n DWORD ?

k DWORD ?

msgone db "Enter n:", 10, 0

msgtwo db "Enter k:", 10, 0

format1 db "%d", 0

format2 db "Total = %d", 10, 0

.CODE

main PROC

;initialization

mov esi, offset x

mov ecx, 4

mov eax, 0

;Calling printf to ask for n

pusha

push offset msgone ;Pushing msgone

call \_printf

add esp, 4 ; restoring the top of stack

popa ;restoring all registers

;Getting input using scanf

pusha ;push all registers

push esi ;address n used for input

push offset format1 ;format

call \_scanf

add esp, 8

popa ;restoring all registers

mov eax, dword ptr [esi] ;Moving n into memory

mov n, eax;

mov esi, offset x ;Setting up for k input

;Calling printf to ask for k

pusha

push offset msgtwo ;pushing msgtwo

call \_printf

add esp, 4

popa ;restoring all registers

;Getting input using scanf

pusha ;pushing all registers

push esi ;parameter k

push offset format1 ;format string

call \_scanf

add esp, 8

popa ;poping all registers

mov ebx, dword ptr [esi] ;moving k into ebx

mov k, ebx; moving k into memory

;Permeutation

mov total, 1 ;setting up total to default contain 1

mov eax, n; move n into eax

mov ebx, k; move k into ebx

cmp eax, ebx

jne fact

mov ecx, 0

mov ebx, n

mov eax, total;

furever: cmp ecx, ebx

je woof

mul ebx; total=total\*n

dec ebx;

mov total, eax;

jmp furever;

fact:

mov ebx, k;

mov eax, 1;

forever: cmp eax, ebx ;condition for while 1<k

jnl endwhile ;jump to endwhile if not less than

dec ebx; k=k-1

mov k, ebx; k into memory

mov eax, n; n into eax

sub eax, ebx; n-k

mov ebx, eax; move result of subtraction into ebx

mov eax, total; move total from memory into eax

mul ebx; total=total\*(n-k)

mov total, eax; new total into memory

mov ebx, k; k back into ebx

mov eax, 1; make eax 1

jmp forever ;jump to while

endwhile:

mov eax, k;

cmp eax, 1; if(k==1)

jne beau

mov eax, total;

mov ebx, n;

mul ebx; total=total\*n

mov total, eax;

beau:

woof:

;mov total, eax ; sum to memory

;outputting using printf

push total ; first paramenter address sum

push offset format2 ; second parameter: format string

call \_printf

add esp, 8 ; restore top of the stack

ret 0

main ENDP

END ; end of source code

### Results:

My results are as follows for the detailed n and k value pairs

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | K VALUES | | | | | | | |  |
| N  V  A  L U E S |  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 1 | 2 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 1 | 3 | 6 | 6 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 1 | 4 | 12 | 24 | 24 | 0 | 0 | 0 | 0 | 0 |
| 5 | 1 | 5 | 20 | 60 | 120 | 120 | 0 | 0 | 0 | 0 |
| 6 | 1 | 6 | 30 | 120 | 360 | 720 | 720 | 0 | 0 | 0 |
| 7 | 1 | 7 | 42 | 210 | 840 | 2520 | 5040 | 5040 | 0 | 0 |
| 8 | 1 | 8 | 56 | 336 | 1680 | 6720 | 20160 | 40320 | 40320 | 0 |
|  | 9 | 1 | 9 | 72 | 504 | 3024 | 15120 | 60480 | 181440 | 362880 | 362880 |