Aufgabe

Lese aus der Datei “f1.txt” eine Sequenz *s* von maximal 8 Charaktern (ohne Leerzeichen).

Berechne die Länge *N* der gelesenen Sequenz und überprüfe ob 1 <= N <= 8. Falls die Länge nicht in dem Intervall enthalten ist, dann wird folgende Fehlermeldung angezeigt: “Die Länge der Sequenz muss zwischen 1 und 8 sein!”.

Lese von der Tastatur eine Zahl *k* zwischen 1 und *N* und überprüfe ob 1<= k <= N. Falls *k* nicht in dem Intervall enthalten ist, dann wird die Fehlermeldung “Falsches k! Gebe die Zahl k wieder ein.” angezeigt und man liest die Zahl k wieder.

Erzeuge alle Cnk Untersequenzen von *s* und drucke sie auf dem Bildschirm, jede auf eine neue Zeile.

Hausaufgabe:

Versuche dann die Untersequenzen:

In einer Datei *f2.txt* zu schreiben.

In derselben Datei *f1.txt* am Ende der Datei zu schreiben.

S: abcdef

N=6

K=3

Auf dem Bildschirm:

abc

abd

abe

abf

acd

..

bcd

…

Idee:

S: a b c d e f

1 1 1 0 0 0 - abc

1 1 0 1 0 0 - abd

0 0 0 0 0 1 - f

0 0 0 0 1 0 - e

….

**1 0 1 1 0 0 - a c d**

…

1 1 1 1 0 1

1 1 1 1 1 0 (5 biti 1)

1 1 1 1 1 1 - abcdef

Implementiere eine Funktion mit folgenden Parametern:

* + die Sequenz **s** – definiert als global
  + die Länge der Sequenz ***N*** – definiert als global
  + **k** gegeben über dem Stack
  + ECX – 1.....1 (N – Bits mit dem Wert 1)

welche alle Untersequenzen erzeugt und auf dem Bildschirm ausdruckt.

Function.asm

|  |
| --- |
| Bits 32  Global Function  Extern s, N  Segment data use32 class=data  Poz db 0  Format db ‘%c’, 0  Enter db 10, 13, 0 ; reprezinta un enter  Segment code use32 class=code  Function:  MyLoop:  Mov EBX, ECX ; ECX = 32 bits din care cei 24 bits high sigur sunt 0  Mov AL, 0  VerificareECXZero:  Shr EBX, 1  Adc AL, 0  Cmp EBX, 0  Je gata  Jmp VerificareECXZero  Gata:  Cmp AL, **byte** [ESP+4] ; compara cu k, ia marimea primului operand pentru al doilea operand  Jne urmatorul  ;afisarea subsirului  ; printf(format, caracter)  Push **ECX**  Identificare\_carcater:  **Shl CL,** 1 ; subsirul e pe CL (citim doar 8 caractere), nu pe ECX  Jnc next  Mov EAX, 0 ; pentru a face caracterul dword pentru afisare  Mov AL, [poz]  Mov EBX, s  Xlat ; AL = caracterul din s corespunzator pozitiei  Push ECX ; CL se strica dupa printf. Avem nevoie de el la cmp  Push dword EAX  Push dword format  Call [printf]  Add esp, 4\*2  Pop ECX  next:  Inc byte[poz]  **Cmp CL, 0**  Jne Identificare\_caracter  Push dword enter  Call [printf]  Add esp, 4  Pop **ECX**  urmatorul:  Loop MyLoop ; decrementeaza ECX si daca n-am ajuns la zero ne intoarcem inapoi  ret |

Obs.

0 0 0 0 1 1 1 1 N=4 , k=4

1 1 1 1 0 0 0 0 (8-N )

Poz= 0 1 2 3 4

If( egale) { ; e mai usor sa ne folosim de jumpul opus pentru ca nu avem else

Afisam }

Decrementan

AL - pozitia elementului din sir

EBX - adresa sirului

XLAT ->efect: AL = elementul din sir

S: ‘abcdef’

AL = 1 -> S+AL

XLAT ;AL = ‘b’

‘ab cd ef’ , AL = 2 -> AL = ‘ ‘

‘ab’,’cd’,’ef’ <=> ‘abcdef’

|  |
| --- |
| Bits 32 Extern Function  Global s,N  Segment data use32 class=data  S times 8 db 0  Format db “%s”,0  N dd 0  K dd 0  Fisier db ‘f1.txt’,0  Access db “r”,0  DescFis dd 0  Segment code use32 class=code  Start:  Push dword Access  Push dword Fisier  Call [fopen]  Add esp,2\*4  Cmp EAX,0  JE Eroare1  Mov [DescFis],EAX  ;fread(buffer, size, count, descriptor) ; count reprezinta cat pot citi maxim  Push dword [DescFis]  Push dword 8  Push dword 1  Push dword S  Call [Fread]  Mov [N],EAX  ADD esp,4\*4  CMP dword [N],0 ; fisierul e gol  JE Eroare2  Repeat:  ;scanf(format,buffer)  Push dword K  Push dword Format  Call [scanf]  ADD Esp,4\*2  CMP dword [K],0  JBE Repeat ; sare doar cand e zero  Mov EAX,[N]  CMP dword [K],EAX  JA Repeat  MOV EBX, 1  Mov ECX,[N]  SHL EBX, CL ;shiftam la stanga cu N pozitii obtinem 2^N  ;100000 - N de 0  SUB EBX,1  ;1....1 - N de 1  Mov ECX, EBX  Push dword [K]  Call Function  ADD esp,4  JMP Final  Eroare1:  ;afisam eroare 1  Jmp Final  Eroare2:  Final:  Push dword 0  Call [exit] |

Hausaufgabe: Quiz Seminar 7 – Wiederholung