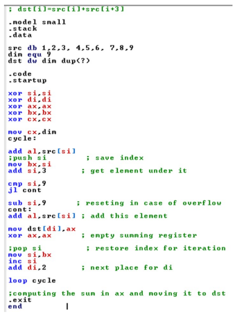
8086

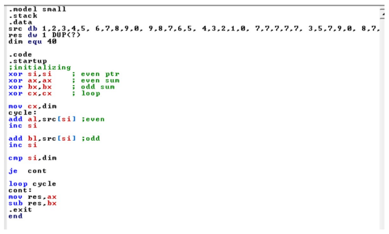
Monday, June 20, 2022 11:35 PM

Cumulative Sum



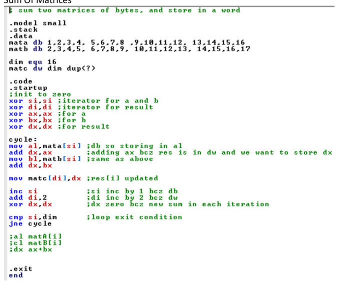
```
; dst[i]=src[i]+src[i+3]

.model small
.stack
.data

src db 1,2,3, 4,5,6, 7,8,9
dim equ 9
dst dw dim dup(?)

.code
.startup

xor si,si
xor di,di
xor ax,ax
xor bx,bx
xor cx,cx

mov cx,dim
cycle:

add al,src[si]
;push si           ; save index
mov bx,si
add si,3          ; get element under it

cmp si,9
jl cont

sub si,9         ; reseting in case of overflow
cont:
add al,src[si] ; add this element

mov dst[di],ax
xor ax,ax         ; empty summing register

;pop si            ; restore index for iteration
mov si,bx
inc si
add di,2          ; next place for di

loop cycle

;computing the sum in ax and moving it to dst
.exit
end
```

Even Indices - Odd Indices



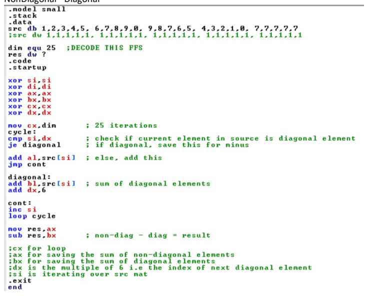
```
.model small
.stack
.data
src db 1,2,3,4,5, 6,7,8,9,0, 9,8,7,6,5, 4,3,2,1,0, 7,7,7,7,7, 3,5,7,9,0, 8,7,
res dw 1 DUP(?)
dim equ 40

.code
.startup
;initializing
xor si,si    ; even ptr
xor ax,ax    ; even sum
xor bx,bx    ; odd sum
xor cx,cx    ; loop

mov cx,dim
cycle:
add al,src[si] ;even
inc si

add bl,src[si] ;odd
inc si

cmp si,dim

je  cont

loop cycle
cont:
mov res,ax
sub res,bx
.exit
end
```

Sum Of Matrices



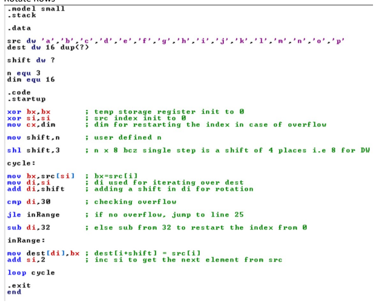
```
; sum two matrices of bytes, and store in a word

.model small
.stack
.data
mata db 1,2,3,4, 5,6,7,8 ,9,10,11,12, 13,14,15,16
matb db 2,3,4,5, 6,7,8,9, 10,11,12,13, 14,15,16,17

dim equ 16
matc dw dim dup(?)

.code
.startup
;init to zero
xor si,si ;iterator for a and b
xor di,di ;iterator for result
xor ax,ax ;for a
xor bx,bx ;for b
xor dx,dx ;for result

cycle:
mov al,mata[si] ;db so storing in al
add dx,ax       ;adding ax bcz res is in dw and we want to store dx
mov bl,matb[si] ;same as above
add dx,bx

mov matc[di],dx ;res[i] updated

inc si          ;si inc by 1 bcz db
add di,2        ;di inc by 2 bcz dw
xor dx,dx       ;dx zero bcz new sum in each iteration

cmp si,dim      ;loop exit condition
jne cycle

;al matA[i]
;cl matB[i]
;dx ax+bx


.exit
end
```

NonDiagonal - Diagonal

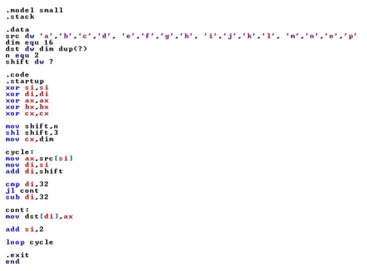


```
.model small
.stack
.data
src db 1,2,3,4,5, 6,7,8,9,0, 9,8,7,6,5, 4,3,2,1,0, 7,7,7,7,7
;src dw 1,1,1,1,1, 1,1,1,1,1, 1,1,1,1,1, 1,1,1,1,1, 1,1,1,1,1

dim equ 25  ;DECODE THIS FFS
res dw ?
.code
.startup

xor si,si
xor di,di
xor ax,ax
xor bx,bx
xor cx,cx
xor dx,dx

mov cx,dim      ; 25 iterations
cycle:
cmp si,dx       ; check if current element in source is diagonal element
je diagonal     ; if diagonal, save this for minus

add al,src[si]  ; else, add this
jmp cont

diagonal:
add bl,src[si]  ; sum of diagonal elements
add dx,6

cont:
inc si
loop cycle

mov res,ax
sub res,bx      ; non-diag - diag = result

;cx for loop
;ax for saving the sum of non-diagonal elements
;bx for saving the sum of diagonal elements
;dx is the multiple of 6 i.e the index of next diagonal element
;si is iterating over src mat
.exit
end
```

Rotate Rows

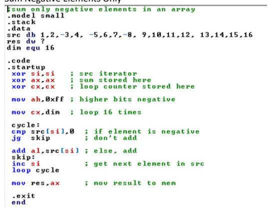


```
.model small
.stack

.data

src dw 'a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p'
dest dw 16 dup(?)

shift dw ?

n equ 3
dim equ 16

.code
.startup

xor bx,bx        ; temp storage register init to 0
xor si,si        ; src index init to 0
mov cx,dim       ; dim for restarting the index in case of overflow

mov shift,n      ; user defined n

shl shift,3      ; n x 8 bcz single step is a shift of 4 places i.e 8 for DW

cycle:

mov bx,src[si]   ; bx=src[i]
mov di,si        ; di used for iterating over dest
add di,shift     ; adding a shift in di for rotation

cmp di,30        ; checking overflow

jle inRange      ; if no overflow, jump to line 25

sub di,32        ; else sub from 32 to restart the index from 0

inRange:

mov dest[di],bx ; dest[i+shift] = src[i]
add si,2         ; inc si to get the next element from src

loop cycle

.exit
end
```

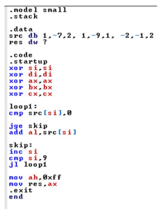


```
.model small
.stack

.data
src dw 'a','b','c','d', 'e','f','g','h', 'i','j','k','l', 'm','n','o','p'
dim equ 16
dst dw dim dup(?)
n equ 2
shift dw ?

.code
.startup
xor si,si
xor di,di
xor ax,ax
xor bx,bx
xor cx,cx

mov shift,n
shl shift,3
mov cx,dim

cycle:
mov ax,src[si]
mov di,si
add di,shift

cmp di,32
jl cont
sub di,32

cont:
mov dst[di],ax

add si,2

loop cycle

.exit
end
```

Sum Negative Elements Only

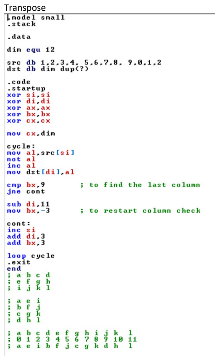


```
;sum only negative elements in an array
.model small
.stack
.data
src db 1,2,-3,4, -5,6,7,-8, 9,10,11,12, 13,14,15,16
res dw ?
dim equ 16

.code
.startup
 xor si,si   ; src iterator
 xor ax,ax   ; sum stored here
 xor cx,cx   ; loop counter stored here

 mov ah,0xff ; higher bits negative

 mov cx,dim  ; loop 16 times

 cycle:
 cmp src[si],0  ; if element is negative
 jg  skip       ; don't add

 add al,src[si] ; else, add
 skip:
 inc si         ; get next element in src
 loop cycle

 mov res,ax     ; mov result to mem

 .exit
 end
```



```
.model small
.stack

.data
src db 1,-7,2, 1,-9,1, -2,-1,2
res dw ?

.code
.startup
xor si,si
xor di,di
xor ax,ax
xor bx,bx
xor cx,cx

loop1:
cmp src[si],0

jge skip
add al,src[si]

skip:
inc si
cmp si,9
jl loop1

mov ah,0xff
mov res,ax
.exit
end
```

Transpose



```
.model small
.stack

.data

dim equ 12

src db 1,2,3,4, 5,6,7,8, 9,0,1,2
dst db dim dup(?)

.code
.startup
xor si,si
xor di,di
xor ax,ax
xor bx,bx
xor cx,cx

mov cx,dim

cycle:
mov al,src[si]
not al
inc al
mov dst[di],al

cmp bx,9        ; to find the last column
jne cont

sub di,11
mov bx,-3       ; to restart column check

cont:
inc si
add di,3
add bx,3

loop cycle
.exit
end
; a b c d
; e f g h
; i j k l

; a e i
; b f j
; c g k
; d h l

; a b c d e f g h i j k  l
; 0 1 2 3 4 5 6 7 8 9 10 11
; a e i b f j c g k d h  l
```

fill like this

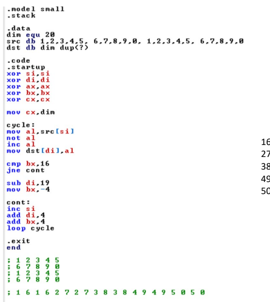
① a _ _ b _ _ c _ _d

② e - f - g -

③ h i j

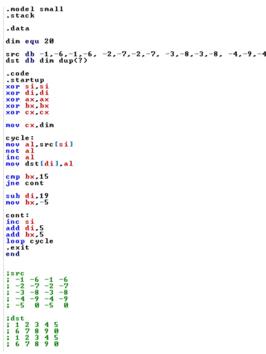
bx finds column

```
3x4    9     11    -3     3     3
4x3    8     11    -4     4     4


a b c d e   a f k p
f g h i j   b g l q
k l m n o   c h m r
p q r s t   e j o t

4x5   16     19    -4     4     4

a b c d     a e i m q
e f g h     b f j n r
i j k l     c g k o s
m n o p     d h l p t
q r s t

5x4   15     19    -5     5     5
```

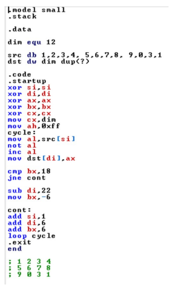


```
.model small
.stack

.data
dim equ 20
src db 1,2,3,4,5, 6,7,8,9,0, 1,2,3,4,5, 6,7,8,9,0
dst db dim dup(?)

.code
.startup
xor si,si
xor di,di
xor ax,ax
xor bx,bx
xor cx,cx

mov cx,dim

cycle:
mov al,src[si]
not al
inc al
mov dst[di],al

cmp bx,16
jne cont

sub di,19
mov bx,-4

cont:
inc si
add di,4
add bx,4
loop cycle

.exit
end

; 1 2 3 4 5
; 6 7 8 9 0
; 1 2 3 4 5
; 6 7 8 9 0

; 1 6 1 6 2 7 2 7 3 8 3 8 4 9 4 9 5 0 5 0
```

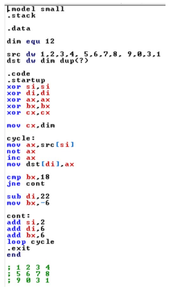
1616
2727
3838
4949
5050



```
.model small
.stack

.data

dim equ 20

src db -1,-6,-1,-6, -2,-7,-2,-7, -3,-8,-3,-8, -4,-9,-4,-9, -5,0,-5,0
dst db dim dup(?)

.code
.startup
xor si,si
xor di,di
xor ax,ax
xor bx,bx
xor cx,cx

mov cx,dim

cycle:
mov al,src[si]
not al
inc al
mov dst[di],al

cmp bx,15
jne cont

sub di,19
mov bx,-5

cont:
inc si
add di,5
add bx,5
loop cycle
.exit
end


;src
; -1 -6 -1 -6
; -2 -7 -2 -7
; -3 -8 -3 -8
; -4 -9 -4 -9
; -5  0 -5  0

;dst
; 1 2 3 4 5
; 6 7 8 9 0
; 1 2 3 4 5
; 6 7 8 9 0
```



```
.model small
.stack

.data

dim equ 12

src db 1,2,3,4, 5,6,7,8, 9,0,3,1
dst dw dim dup(?)

.code
.startup
xor si,si
xor di,di
xor ax,ax
xor bx,bx
xor cx,cx
mov cx,dim
mov ah,0xff
cycle:
mov al,src[si]
not al
inc al
mov dst[di],ax

cmp bx,18
jne cont

sub di,22
mov bx,-6

cont:
add si,1
add di,6
add bx,6
loop cycle
.exit
end

; 1 2 3 4
; 5 6 7 8
; 9 0 3 1
```



```
.model small
.stack

.data

dim equ 12

src dw 1,2,3,4, 5,6,7,8, 9,0,3,1
dst dw dim dup(?)

.code
.startup
xor si,si
xor di,di
xor ax,ax
xor bx,bx
xor cx,cx

mov cx,dim

cycle:
mov ax,src[si]
not ax
inc ax
mov dst[di],ax

cmp bx,18
jne cont

sub di,22
mov bx,-6

cont:
add si,2
add di,6
add bx,6
loop cycle
.exit
end

; 1 2 3 4
; 5 6 7 8
; 9 0 3 1
```