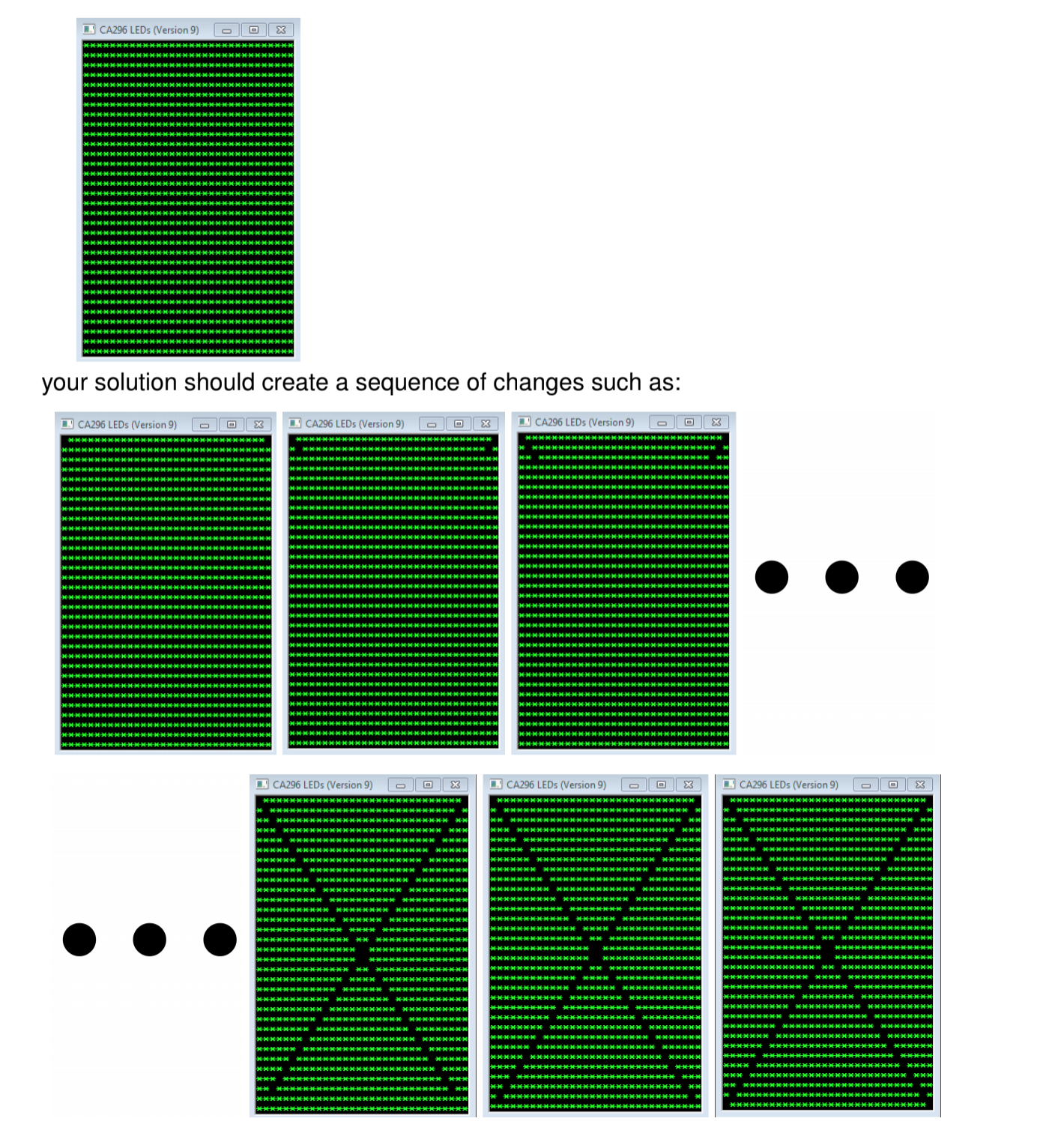
2018:

[20 marks]

Assuming that the LEDs have been set to some pattern, write MASM code to switch off the LEDs on the diagonals. Your code should process one row at a time.

For example, given the initial pattern:

;

; Data & Code

;

.data

row\_num DWORD 0

mask1 DWORD 080000000h ; from left hand side

mask2 DWORD 01h ; from right hand side

get\_patt BYTE 'Please enter a pattern number: ',0,10

message BYTE 'Press OK to continue',0

caption BYTE 'Welcome to CA296',0

.code

main:nop

invoke version

invoke readIntegerWithMessage, ADDR get\_patt

invoke setPattern, eax

start:

cmp row\_num, 32

je finish

invoke readRow, row\_num ; row pattern is stored in eax

from\_left:

mov ebx, mask1 ; put the mask1 into ebx because we edit it

not ebx ; not mask1 (011111111..)

and eax, ebx ; and our row to the not-ed mask1

from\_right:

mov edx, mask2 ; put mask2 into edx because we edit it

not edx ; not mask2 (1111111110)

and eax, edx ; and our row to not-ed mask2

turn\_off:

invoke writeRow, row\_num, eax

reset:

shr mask1, 1

shl mask2, 1

invoke Sleep, 100

inc row\_num

jmp start

finish:

invoke MessageBox,0,ADDR message,ADDR caption,MB\_OK

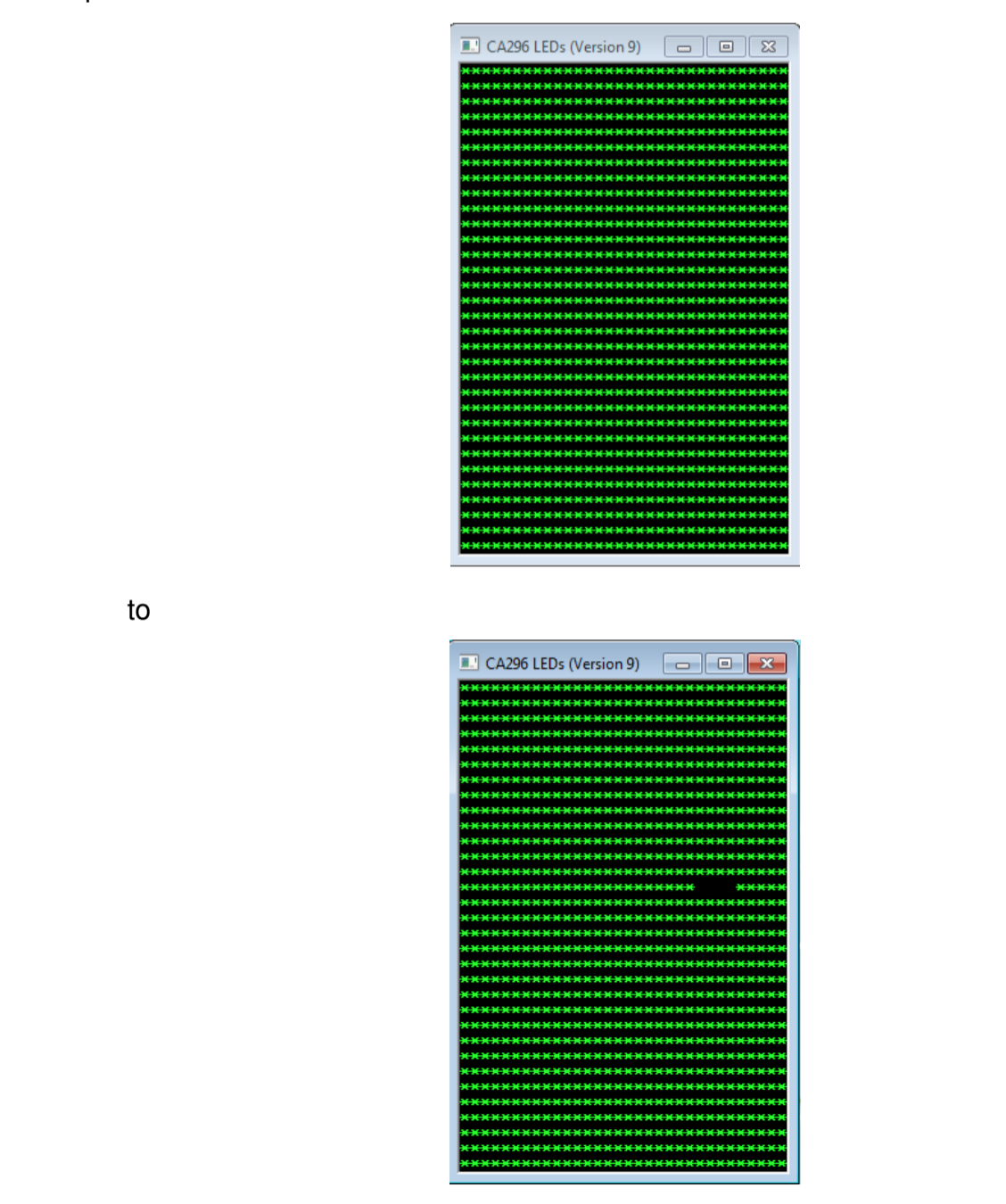
invoke ExitProcess,0

end main

When you ‘and’ something you only get a 1 if there are two ones, otherwise it is a zero. Here, because we want to switch lights off, we made our mask be a zero so when we ‘and’ it to the pattern, it can’t return a 1, therefore switches off. (See google doc title ‘Helpful Explanations for Logical Instructions’ for more)

2017:

[ 20 Marks ]

Given the definitions

.data

row DWORD ?

start DWORD ?

finish DWORD ?

assuming that the values of row, start and finish are each in the interval [0..31], write

MASM code to switch off the LEDs in the interval [start..finish] on row row of the

displayed LED pattern.

For example, if row is 13, start is 5 and finish is 8, your code should change the

pattern

;

; Data & Code

;

.data

row\_num DWORD 0

start\_point DWORD 0

end\_point DWORD 0

mask1 DWORD 01h

count DWORD 0

get\_patt BYTE 'Please enter a pattern number: ', 0

get\_row BYTE 'Please enter a row number to edit: ',0

get\_start BYTE 'Please enter a column off start number: ',0

get\_end BYTE 'Please enter a column off end number: ',0

message BYTE 'Press OK to continue',0

caption BYTE 'Welcome to CA296',0

.code

main:nop

invoke version

setup:

invoke readIntegerWithMessage, ADDR get\_row

mov row\_num, eax

invoke readIntegerWithMessage, ADDR get\_start

mov start\_point, eax

invoke readIntegerWithMessage, ADDR get\_end

mov end\_point, eax

invoke readIntegerWithMessage, ADDR get\_patt

invoke setPattern, eax

mov ecx, 0 ; my counter (needs to be register to compare)

set\_mask:

cmp ecx, start\_point ; compare counter to start point

je start

shl mask1, 1

inc ecx ; increase counter

jmp set\_mask

start:

cmp ecx, end\_point ; compare counter with end point

jg finish ; when greater, we are done

mov count, ecx ; move ecx into counter because ‘invoke’ resets registers

invoke readRow, row\_num ; read in the chosen row (then stored in eax)

mov ebx, mask1 ; move mask into ebx (to edit we put in a register)

not ebx ; puts 0s where 1s and 1s where 0s

and eax, ebx ; only get a 1 if two ones

invoke writeRow, row\_num, eax

reset:

shl mask1, 1

invoke Sleep, 200

mov ecx, count ; puts count back into ecx

inc ecx ; increase counter

jmp start ; go around again

finish:

invoke MessageBox,0,ADDR message,ADDR caption,MB\_OK

invoke ExitProcess,0

end main

2017R:

[ 20 Marks ]

; Given the definitions

; .data

; column DWORD ?

; start DWORD ?

; finish DWORD ?

; assuming that the values column, start and finish are each in the interval [0...31],

; write MASM code to switch off the LEDs in the interval [start..finish] on column

; column of the displayed LED pattern.

; For example, if column is 5, start is 5 and finish is 8, your code should change the

; pattern:

.data

col DWORD 0

start\_p DWORD 0

end\_p DWORD 0

count DWORD 0

mask1 DWORD 01h

get\_start BYTE 'Please enter a start point',0

get\_end BYTE 'Please enter an end point',0

get\_col BYTE 'Please enter a column number',0

get\_patt BYTE 'Please enter a pattern number',0

message BYTE 'Press OK to continue',0

caption BYTE 'Welcome to CA296',0

.code

main:nop

setup:

invoke readIntegerWithMessage, ADDR get\_col

mov col, eax

invoke readIntegerWithMessage, ADDR get\_start

mov start\_p, eax

invoke readIntegerWithMessage, ADDR get\_end

mov end\_p, eax

invoke readIntegerWithMessage, ADDR get\_patt

invoke setPattern, eax

move\_mask:

mov eax, mask1 ; put mask into eax

sal eax, 1 ; move mask (in eax) left 1

mov mask1, eax ; put edited mask into mask variable

inc count ; increase count with each edit

mov edx, count ; put our count into edx (we need one register when comparing)

cmp edx, col ; check if we are in the right place

jl move\_mask ; if less, go around again

; if more go to next step

lights\_off:

mov eax, end\_p ; put our end point into eax

cmp start\_p, eax ; check where we are

jg finish ; we are done when greater

invoke readRow, start\_p ; read first row in (stored in eax)

not eax ; turns 111 to 000

mov ebx, mask1 ; puts our mask into ebx

or eax, ebx ; forces a 1 to be present

; (you can only get a 0 if two 0s, but our mask is 1)

not eax ; turns our row back to original plus mask

invoke writeRow, start\_p, eax

invoke Sleep, 200

inc start\_p ; to get to next row

jmp lights\_off

finish:

invoke MessageBox,0,ADDR message,ADDR caption,MB\_OK

invoke ExitProcess,0

end main

2016:

[ 20 Marks ]

; Given a 32-bit value we can swap its upper and lower 16 bits. For example, given

; 0000 0001 0010 0011 0100 0101 0110 0111

; we can swap its upper and lower 16 bits to get the value:

; 0100 0101 0110 0111 0000 0001 0010 0011

; Write MSAM code to input an integer N in the interval [0...5] and output the LED pattern

; for N with the upper and lower 16 bits of the value for each row swapped. For example,

; given the initial pattern

.data

row\_num DWORD 0

left DWORD 0

right DWORD 0

row DWORD 0

get\_patt BYTE 'Please enter a pattern number',0

message BYTE 'Press OK to continue',0

caption BYTE 'Welcome to CA296',0

.code

main:nop

invoke version

invoke readIntegerWithMessage, ADDR get\_patt

invoke setPattern, eax

swap\_patt:

cmp row\_num, 32 ; at the end of the rows we are done

je finish

invoke readRow, row\_num ; read this row (stored in eax)

mov row, eax ; put eax into variable called row

mov ecx, 16 ; put 16 into ecx (then we can call 16 using CL)

shl eax, CL ; shift row left by 16

mov left, eax ; put new row into left

mov eax, row ; move original row into eax

shr eax, CL ; shift right by 16

mov right, eax ; move edited row into right

add eax, left ; add right(eax) and left (answer stored in eax)

invoke writeRow, row\_num, eax

invoke Sleep, 200

inc row\_num ; next row

jmp swap\_patt ; go around again

finish:

invoke MessageBox,0,ADDR message,ADDR caption,MB\_OK

invoke ExitProcess,0

end main

2016R:

[ 20 Marks ]

;

; Write MASM code to input an integer N in the interval [0..5] and output the LED pattern

; for N in which only the LEDs that are part of a vertical line from top to bottom in the

; original pattern are left switched on. For example, given the initial pattern

;

.data

on\_count DWORD 0

row\_num DWORD 0

col\_num DWORD 0

mask1 DWORD 0ffffffffh ; we start our mask off with all 1s

get\_patt BYTE 'Please enter a pattern number',0

message BYTE 'Press OK to continue',0

caption BYTE 'Welcome to CA296',0

.code

main:nop

invoke version

invoke readIntegerWithMessage, ADDR get\_patt

invoke setPattern, eax

make\_mask:

cmp row\_num, 32 ; see how far we are

je reset ; if done go to next step

invoke readRow, row\_num ; read in the row (stored in eax)

and eax, mask1 ; and our mask (only get 1 if two 1s)

mov mask1, eax ; put result into mask to replace old mask

inc row\_num ; next row

jmp make\_mask

reset:

mov row\_num, 0 ; reset row\_num

write\_patt:

cmp row\_num, 32 ; see how far we are

je finish ; if 32 - done

invoke writeRow, row\_num, mask1 ; each row the same as our latest mask

invoke Sleep, 200

inc row\_num

jmp write\_patt ; go around again

finish:

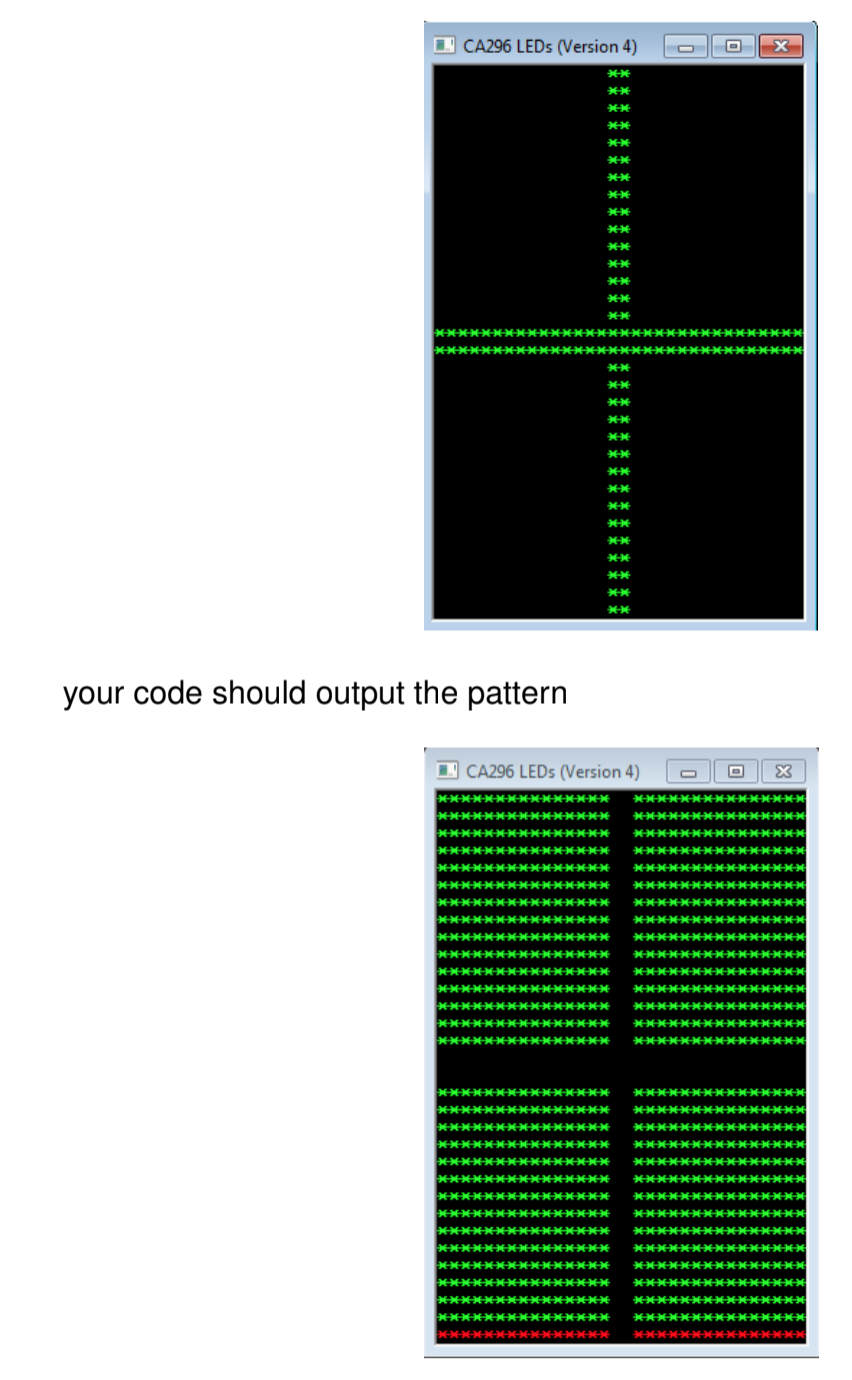
invoke MessageBox,0,ADDR message,ADDR caption,MB\_OK

invoke ExitProcess,0

end main

2016 SAMPLE EXAM:

[ 20 Marks ]

; Write MASM code to input an integer N in the interval [0..5] and output the LED pattern

; for N with all the LEDs inverted. For example, given the initial pattern

.data

row\_num DWORD 0

get\_patt BYTE 'Please enter a pattern number',0

message BYTE 'Press OK to continue',0

caption BYTE 'Welcome to CA296',0

.code

main:nop

set\_up:

invoke version

invoke readIntegerWithMessage, ADDR get\_patt

invoke setPattern, eax

invert:

cmp row\_num, 32

je finish

invoke readRow, row\_num ; stored in eax

not eax ; inverts the row

invoke writeRow, row\_num, eax ; print inverted row

invoke Sleep, 200

inc row\_num ; next row

jmp invert ; go around again

finish:

invoke MessageBox,0,ADDR message,ADDR caption,MB\_OK

invoke ExitProcess,0

end main