פעולות אריתמטיות לוגיות

# פקודות לוגיות

פקודות לוגיות הן שימושיות מאוד כשאנחנו רוצים לשנות את ערכו של ביט, או מספר ביטים, ובאותו זמן להשאיר ערכים של ביטים אחרים בלי שינוי. פעולה זו נקראת מיסוך MASKING.

למה בכלל נרצה להתעסק עם ביטים בודדים?

**הצפנות**, לדוגמה. כשמצפינים מידע, המידע שמיועד להצפנה נשמר בצורה "דחוסה" – Packed data. נסתכל לדוגמה על מערך של שמונה בתים, כל אחד מהם שומר ערך שהוא 0 או אחד:

00000000 00000001 00000001 00000001 00000000 00000000 00000001 00000000

את אותו המידע אפשר לשמור בבית אחד בצורה דחוסה:

01110010

בצורה דחוסה אנחנו יכולים להאיץ את מהירות ההצפנה והפענוח, יחסית למהירות שניתן להשיג כשעובדים בצורה לא דחוסה בה מוקצה בית נפרד לשמירת כל ביט .

על הבית הזה מבצעים פעולות לוגיות שונות שהופכות אותו למוצפן. לדוגמה, אנחנו יכולים להפוך את ערכו של כל ביט שני ,ולקבל:

00100101

מי שלא יודע מה הפעולה שעשינו, לא יוכל לשחזר את הערכים המקוריים ולפענח את המסר המוצפן. בשביל לבצע פעולה כזו, צריך לדעת "לשחק" עם ביטים בודדים. כעת נראה איך עושים את זה.

צפייה ברגיסטרים / משתנים

Alt + v

Registers

Variables

הרצה

tasm /zi file\_name

tlink /v file\_name

td fole name

קיימות ארבע פקודות לוגיות – not ,xor ,or ,and. מיד נסביר מה עושה כל פקודה.

# פקודת and

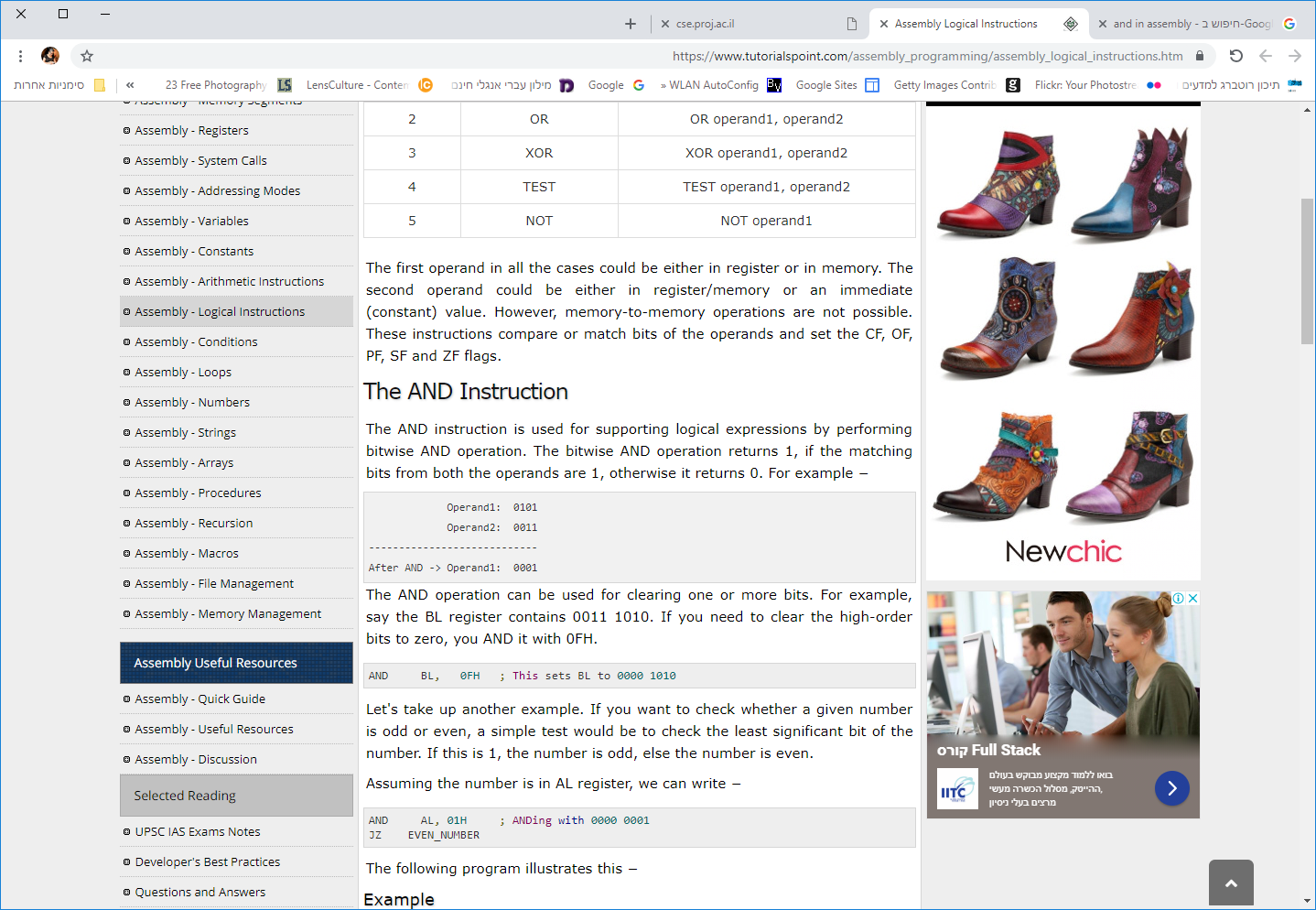
אמת \* אמת ⇦ אמת (כל אפשרות אחת התוצאה היא שקר)

פעולות על ביטים בודדים

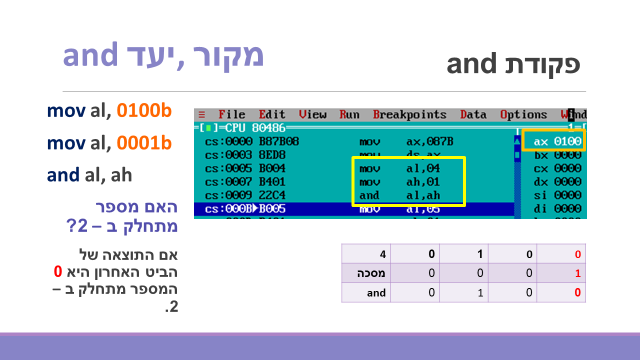
and זו בעצם פעולת הכפלה, רק 1\*1 = 1.

לשים מסכה שבודקת רק את הסיבות הדומות ⇦ פעולה המאפשרת לבודד סיביות

לאפס ביטים







למסכה בדוגמה כאשר ah שווה 1 (0001) אפשר לבדוק האם המספר מתחלק ב – 2.

רק כאשר במספר המקורי (המספר לבדיקה) הספרה האחרונה היא 1 ⇦ נקבל תוצאה של 1 ולכן המספר אי זוגי.



פעולת and יכולה לקבוע לנו את גודל המספר. שמהמספר לא יהיה גדול מ – על פי המסכה שנקבע.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 255 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | הגבלה למספר 15 |

עם המסכה הזו כל מספר שיהי גדול מ – 15 יקבל את הערך 0 ולכן נקבל רק מספרים הקטנים או שווים ל – 15

# פקודת or

פעולת חיבור בני שני אופרנדים רק 0 + 0 ⇦ 0

כל חיבור אחר ייתן את התוצאה 1

נועד לקבע את הערך של ביט אחד או יותר

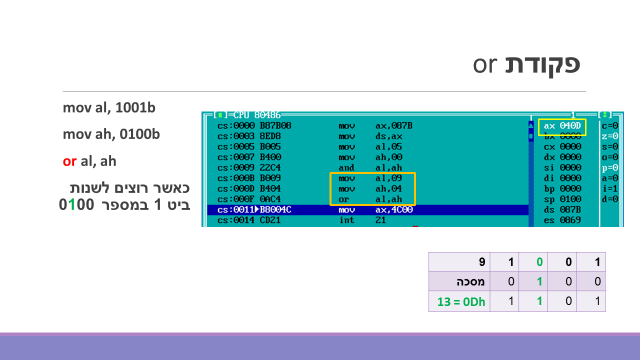
בדוגמה שינוי של 4 הביטים התחתונים לערך 1.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |



פקודה זו מקבלת שני אופרנדים, מבצעת ביניהם חיבור לוגי (OR) סיבית אחרי סיבית, ושומרת את התוצאה באופרנד היעד.

שימוש מקובל בפקודה הוא להעלות ביט אחד בבית ל-1. (מעין הרמת דגל).



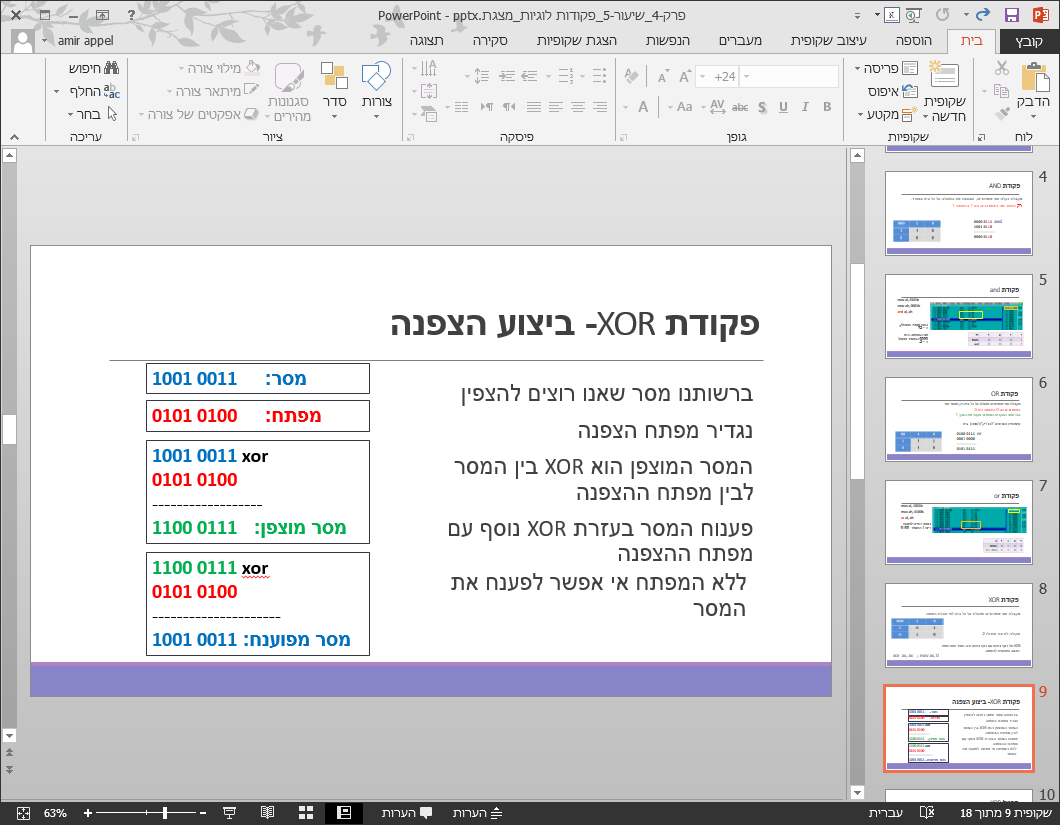
# פקודת xor

נועד לאיפוס רגיסטרים או לצורף פעולת הצפנה

רק אם שני האופרנדים הם דומים נקבל את הערך 1

אם הם שונים נקבל את הערך 0





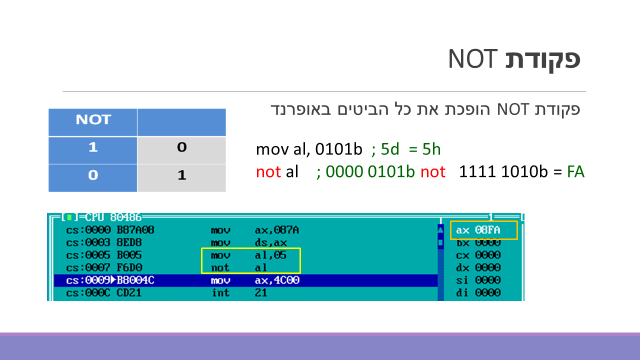
**פקודת XOR- ביצוע הצפנה**

* ברשותנו מסר שאנו רוצים להצפין
* נגדיר מפתח הצפנה
* המסר המוצפן הוא XOR בין המסר לבין מפתח ההצפנה
* פענוח המסר בעזרת XOR נוסף עם מפתח ההצפנה

ללא המפתח אי אפשר לפענח את המסר

# פקודת not

הופכת את הביטים



# פקודות הזזה shl, shr

מקבלות אופרנד ו"מזיזות" את הביטים שלו

shl / shr register, const

shl / shr register, cl

shl / shr memory, const

shl / shr memory, cl

SHR- Shift Right

SHL- Shift Left

שימושים:

כפל וחילוק

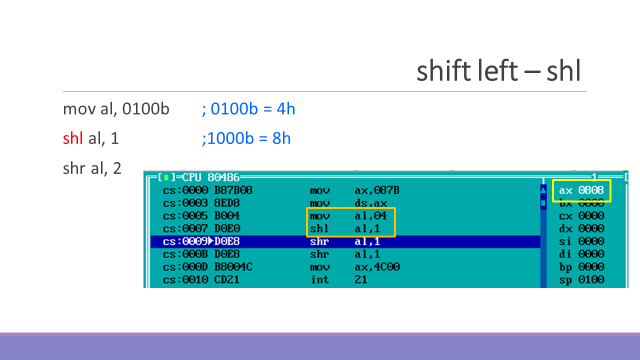
תיקון שגיאות והצפנה

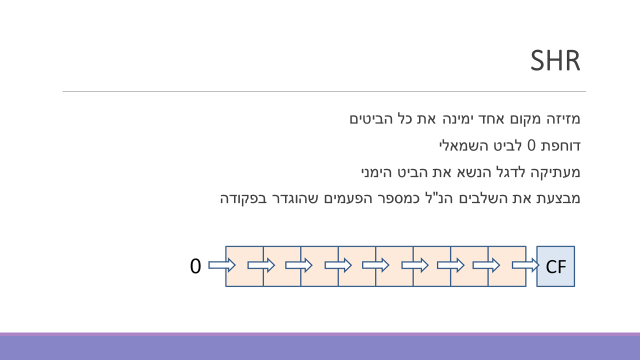
דחיסה ופריסה של מידע

כתיבה לזיכרון הוידאו

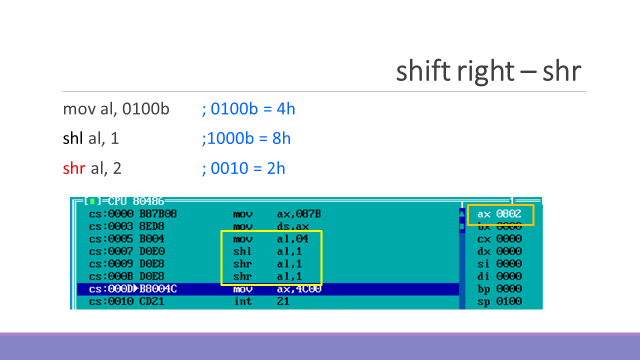


הזזה של הביטים שמאלה והוספת 0 בביט השמאלי ביותר ⇦ **פעולת כפל ב – 2.**





מזיזים את הביטים מקום אחד ימינה, ומוסיפים לביט השמאלי ביותר 0 ⇦ חילוק ב - 2



# תרגול

**פעולות לוגיות**

פתחו את הקובץ base.asm ושימרו אותו בשם אחר.

## קלט ופלט

בתרגיל הבא מספר שאלות שעבורן נצטרך לקלוט ולהדפיס מספרים. לפני שנתחיל לעבוד עם הפקודת הלוגיות צרו את הפרוצדורות הבאות:

* readUserInput
* printString
* printNumber
* newline

|  |  |
| --- | --- |
| ; new line  proc newLine  pusha  mov dl, 0ah  mov ah, 2h  int 21h  popa  ret  endp newLine | ;proc to get input to al  proc readUserInput  ; get user first input number to al  mov ah, 1h  int 21h  sub al, '0'  ret  endp readUserInput |
| ;proc to print the a number stored in al  proc printNum  pusha  mov ah, 0  div [ten]  add ax, '00'  mov dx, ax  mov ah, 2h  int 21h  mov dl, dh  int 21h  popa  ret  endp printNum | ; print message  proc printString  pusha  mov ah, 9h  int 21h  popa  ret  endp printString |

## פקודת and

בעזרת פקודת and נבדוק האם מספר הוא זוגי או אי זוגי

כדי לדעת אם מספר זוגי עלינו לבדקו את הביט הימני ביותר שלו.

מסכה של and:

1 and 1 ⇨ 1

1 and 0 ⇨ 0

0 and 0 ⇨ 0

**הדרכה:**

1. הגדירו שלושה משתנים של מחרוזות

msg db 'Please enter a number: $'

הגדיר עוד שתי 2 מחרוזות odd ו even.   
אם המספר זוגי נדפיס למסך even ואם הוא אי זוגי נדפיס למסך odd

הדפיסו את המחרוזת של בקשה להקליד למסך תו אחד בעזרת הפרוצדורה **printString**  
אל תשכחו להעביר את תחילת המחרוזת לרגיסטר dx ⇦ mov dx, offset msg

1. בעזרת הפרוצדורה readUserInput קלטו מספר כלשהו לרגיסטר al. לדוגמא:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |

al =5 ⇨

1. חשבו מהי המסכה המתאימה ביותר לבדוק האם המספר מתחלק ב – 2. הפעילו את ההוראה and על רגיסטר al ובדקו את התוצאה.  
   אם המספר שווה ל – 0 קפצו לתווית והדפיסו את המחרוזת even

אם המספר אינו שווה לאפס קפצו תווית אחרת והדפיס ולמסך את המחרוזת את odd

1. כתבו תכנית חדשה הבודקת האם מספר (8 ביט) מתחלק ב- 4?

כתבו תכנית הקולטת מספר,   
ובודקת אם הוא מתחלק ב – 4?

**מהי המסכה שצריך לצורך הבדיקה?**

IDEAL

MODEL small

STACK 100h

p186

DATASEG

; --------------------------

entr db "please enter a number: $"

zug db "zugi$"

ezug db "ezug$"

ten db 10

; --------------------------

CODESEG

proc printString

pusha

mov ah, 9h

int 21h

popa

ret

endp printString

proc readUserInput

; get user first input number to al

mov ah, 1h

int 21h

sub al, '0'

ret

endp readUserInput

proc newline

; new line

pusha

mov dl, 0ah

mov ah, 2h

int 21h

popa

ret

endp newLine

proc printNumber

pusha

; print the result stored in AL

mov ah, 0

div [ten]

add ax, '00'

mov dx, ax

mov ah, 2h

int 21h

mov dl, dh

int 21h

call newLine

popa

ret

endp printNumber

DATASEG

; --------------------------

entr db "please enter a number: $"

zug db "zugi$"

ezug db "ezug$"

ten db 10

start:

mov ax, @data

mov ds, ax

; --------------------------

mov dx, offset entr

call printString

call readUserInput

and al, 1

cmp al, 1

je ezugi

zugi:

mov dx, offset zug

jmp print

ezugi:

mov dx, offset ezug

print:

call newLine

call printString

בדיקה באמצעות פקודת AND, אם מספר (8 ביט) מתחלק ב-4?

DATASEG

; --------------------------

entr db "please enter a number: $"

is4 db "mitchalek$"

no4 db "lo mitchalek$"

ten db 10

my\_bits db 0

; --------------------------

start:

mov ax, @data

mov ds, ax

; --------------------------

mov dx, offset entr

call printString

call readUserInput

mov [my\_bits], al

and [my\_bits], 11b

cmp [my\_bits], 0

je div4

not4:

mov dx, offset no4

jmp print

div4:

mov dx, offset is4

print:

call newLine

call printString

## פקודת or

הגדירו משתנה בשם my\_bits בגודל בית וקראו מספר לתוכו.

"הדליקו" את ערכו של הביט השלישי (משמאל) במשתנה, כלומר ודאו שערכו 1. בדקו את התוצאה ב tùrbo debugger

DATASEG

; --------------------------

entr db "please enter a number: $"

my\_bits db 0

; --------------------------

start:

mov ax, @data

mov ds, ax

; --------------------------

mov dx, offset entr

call printString

call readUserInput

mov [my\_bits], al

or [my\_bits], 00100000b

## פקודת xor

קראו ערך כלשהו ל al ואפסו אותו באמצעות פעולת ביטים. הדפיסו את התוצאה

DATASEG

; --------------------------

entr db "please enter a number: $"

ten db 10

; --------------------------

start:

mov ax, @data

mov ds, ax

; --------------------------

mov dx, offset entr

call printString

call readUserInput

xor al, al

call newLine

call printNumber

; --------------------------

## שימוש בפקודות הזזה (shift) לכפל וחילוק

* הכניסו ל-al את הערך 3. בעזרת פקודות הזזה, כיפלו את al ב-4. הדפיסו את התוצאה
* הכניסו ל-al את הערך 136d. בעזרת פקודות הזזה, חלקו את al ב-8. הדפיסו את התוצאה
* הכניסו ל-al את הערך 4 (דצימלי). בעזרת פקודות הזזה וחיבור, כיפלו את al ב-20. הדפיסו את התוצאה

הדרכה: התייחסו ל - 20 בתור סכום של 16 ו-4. השתמשו ברגיסטרים נוספים כדי לשמור חישובי ביניים.

DATASEG

; --------------------------

entr db "please enter a number: $"

ten db 10

; --------------------------

start:

mov ax, @data

mov ds, ax

; --------------------------

mov al, 3

shl al, 2

call printNumber ; 3 \* 4 = 12

mov al, 136d

shr al, 3 ; 136 / (23 ⇨ 8) = 17

call newLine

call printNumber

mov al, 4

mov bl, al

shl bl, 4 ; 4 \* 4

shl al, 2 ; 4 \* 2

add al, bl ; (4 \* 2) + (4 \* 4)

call printNumber



## דף תרגילים ישן --------------------------------------

## פקודת AND

1. העבירו ערך כלשהו ל al. לדוגמא:

mov al, 00110101b

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 |

Al =

כתבו תכנית קצרה המציגה בעזרת פעולות לוגיות, אם המספר זוגי או אי זוגי.

כדי לדעת אם מספר זוגי עלינו לבדקו את הבין הימני ביותר שלו.

מסכה של and:

1 and 1 ⇨ 1

1 and 0 ⇨ 0

0 and 0 ⇨ 0

1. איך אפשר לבדוק באמצעות פקודת AND, אם מספר (8 ביט) מתחלק ב-4?

כתבו תכנית המגדירה משתנה בגודל בית בשם my\_bits, מעבירה לו ערך ובודקת אם הוא מתחלק ב – 4?

מהי המסכה שצריך לצורך הבדיקה?

## פקודת or

הגדירו משתנה בשם my\_bits בגודל בית ושימו בו ערך כלשהו.

"הדליקו" את ערכו של הביט השלישי (משמאל) במשתנה, כלומר ודאו שערכו 1.

## xor

הכנס ערך כלשהו ל al ואפס אותו באמצעות פעולת ביטים.

השוו את אורך הפקודה לאיפוס בעזרת פקודת mov, איזו פקודה קצרה יותר?

## שימוש בפקודות הזזה (shift) לכפל וחילוק

* הכניסו ל-al את הערך 3. בעזרת פקודות הזזה, כיפלו את al ב-4.
* הכניסו ל-al את הערך 136d (88h). בעזרת פקודות הזזה, חלקו את al ב-8.
* הכניסו ל-al את הערך 11 (דצימלי). בעזרת פקודות הזזה וחיבור, כיפלו את al ב-20.

הדרכה: התייחסו ל-20 בתור סכום של 16 ו-4. השתמשו ברגיסטרים נוספים כדי לשמור חישובי ביניים.

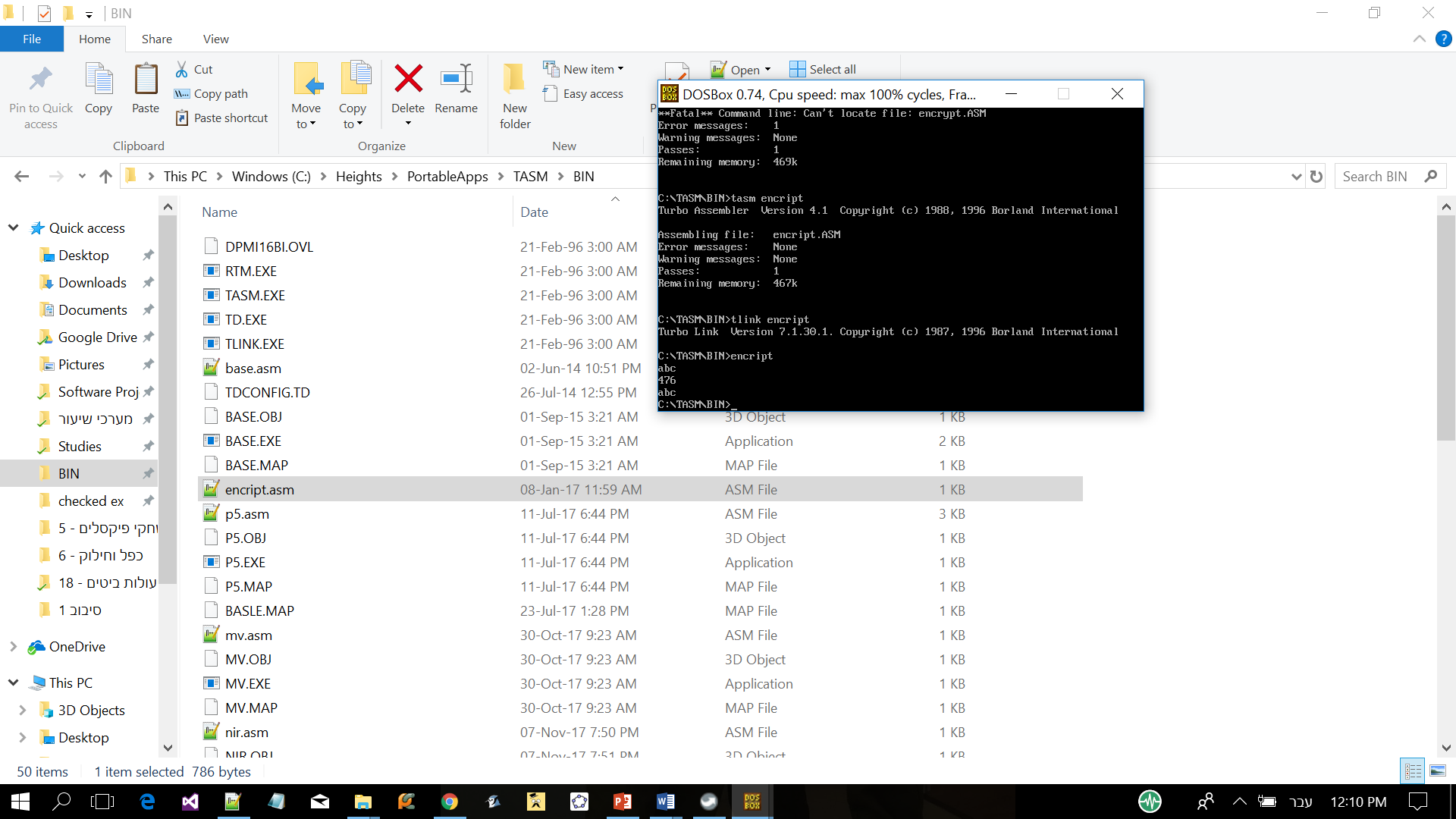
## הצפנה

הגדירו מחרוזת בת 3 אותיות בשם msg שהיא אוסף של תווי ASCII.

לדוגמה ''ABC$, (תו ה - $ משמש להדפסת לסימון סוף מחרוזת)

הגדירו מפתח הצפנה בן 8 ביטים, לבחירתכם.

1. הדפיסו את המסר
2. הצפינו את המסר בעזרת מפתח ההצפנה
3. הדפיסו את המסר המוצפן
4. פענחו את המסר
5. הדפיסו את המסר המפוענח

; print msg

mov dx, offset msg

mov ah, 9h

int 21h

; new line

mov dl, 0ah

mov ah, 2h

int 21h

תשובות

**פקודת and**

1. נתון מספר בן בית אחד היושב ב al. לדוגמא:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 |

חשבו ומצאו באמצעות פעולת ביטים, אם המספר זוגי או אי זוגי.

כדי לדעת אם מספר זוגי עלינו לבדקו את הבין הימני ביותר שלו.

מסכה של and:

1 and 1 ⇨ 1

1 and 0 ⇨ 0

0 and 0 ⇨ 0

and al, 1

מסכה של 00000001 תבדוק אם המספר זוגי ⇦ כי רק 1\*1 = 1

1. איך אפשר לבדוק באמצעות פקודת AND, אם מספר (8 ביט) מתחלק ב-4?

כתבו תכנית המגדירה משתנה בגודל בית בשם my\_bits, מעבירה לו ערך ובודקת אם הוא מתחלק ב – 4?

מהי המסכה שצריך לצורך הבדיקה?

and al, 11

מסכה 00000011 בפעולת and, אם המספר מתחלק ב – 4 התוצאה ברגיסטר תהיה 0000 0000.

כל תוצאה אחת מעידה על כך שהמספר אינו מתחלק ב – 4.

(1 and 1 = 1) מספר המתחלק ב - 4 שני הביטים הנמוכים הם 00.

איך אפשר לבדוק באמצעות פקודת AND:

* אם מספר (8 ביט) (8 ביט) מתחלק ב-4?
* אם מספר signed (8 ביט) הוא שלילי?

כתבו תכנית הבודקת אם מספר מתחלק ב – 4

CODESEG

start:

mov ax, @data

mov ds, ax

; --------------------------

mov cl, 00001100b

mov al, cl

mov bl, 00000011b ; mask

and al, bl

; --------------------------

**פקודת or**

נסו לשנות את ערכו של הביט השלישי (מימין) במספר בינארי 8bit.

DATASEG

my\_bits db 1000b

; --------------------------

CODESEG

start:

mov ax, @data

mov ds, ax

; --------------------------

mov al, 0**1**00b ; The Mask

or [my\_bits], al

**תרגיל XOR**

נתון מספר בן בית אחד היושב ב al. לדוגמא

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 |

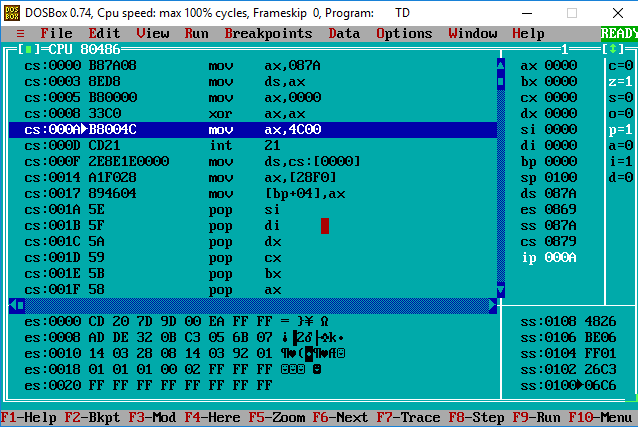
אפס את הרגיסטר al באמצעות פעולת ביטים.

xor al, al

הסבירו מדוע התוצאה של xor של אוסף ביטים עם אוסף ביטים זהה – היא תמיד אפס.

בדקו את שפת המכונה של xor ax, ax ו – mov ax, 0. איזו פקודה קצרה יותר?

xor מקבל ערך של 0 כאשר שני האופרנדים שווים וערך של 1 כאשר הם שונים.



פקודת mov 3 בתים, פקודת xor שני בתים.

**שימוש בפקודות הזזה (shift) לכפל וחילוק**

* הכניסו ל-al את הערך 3. בעזרת פקודות הזזה, כיפלו את al ב-4.
* הכניסו ל-al את הערך 136d (88h). בעזרת פקודות הזזה, חלקו את al ב-8.
* הכניסו ל-al את הערך 4 (דצימלי). בעזרת פקודות הזזה וחיבור, שנו את ערכו ל – 22.

הדרכה: התייחסו ל-22 בתור סכום של 20 ו-2.

mov al, 3

shl al, 2 ; 3 \* 4

mov al, 136

shr al, 3 ; 88h / 8 = 136 / 8

CODESEG

start:

mov ax, @data

mov ds, ax

4 \* 4 + 2

; --------------------------

mov al, 4

**shl al,** **2**

**add** **al, 2**

; --------------------------

exit:

mov ax, 4c00h

int 21h

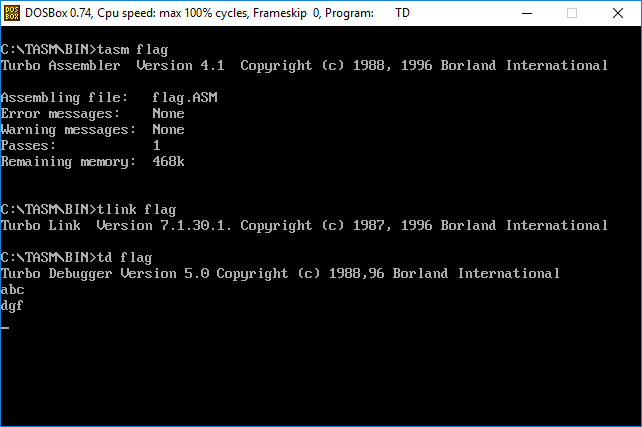
END start

**הצפנה**

הגדירו מחרוזת בת שלושה תווים בשם msg שהיא אוסף של תווי ASCII.

לדוגמה 'ABC $'. (תו ה-$ משמש להדפסת המחרוזת)

הגדירו מפתח הצפנה בן 8 ביטים, לבחירתכם.

1. הדפיסו את המסר
2. הצפינו את המסר בעזרת מפתח ההצפנה
3. הדפיסו את המסר המוצפן
4. פענחו את המסר
5. הדפיסו את המסר המפוענח

DATASEG

msg db 'abc$'

key db 01010101b

CODESEG

start:

mov ax, @data

mov ds, ax

; --------------------------

; print msg

mov dx, offset msg

mov ah, 9h

int 21h

; new line

mov dl, 0ah

mov ah, 2h

int 21h

; הצפנה

mov al, [key]

xor [msg] , al

xor [msg + 1] , al

xor [msg + 2] , al

; print msg

mov dx, offset msg

mov ah, 9h

int 21h

mov dl, 0ah

; new line

mov ah, 2h

int 21h

; --------------------------

exit:

mov ax, 4c00h

int 21h

END start

; print msg

mov dx, offset msg

mov ah, 9h

int 21h

; new line

mov dl, 0ah

mov ah, 2h

int 21h

; פענוח

mov bx, offset msg

mov al, [key]

xor [bx], al

inc bx

xor [bx], al

inc bx

xor [bx], al