

ארגון המחשב ושפת סף (203.1130)

סמסטר ב' תשס"ד
בחינה סופית - מועד ב'

הוראות לנבחן:

- משך הבחינה שלוש שעות.
- מותר להשתמש בכל חומר עזר, למעט מחשבים ומחשבוניס מכל סוג.
- יש להשיב על כל השאלות.
- יש לרשום את התשובות בגוף השאלון במקומות המיועדים לכך.
- נא לכתוב בכתב יד ברור ונקי. מומלץ להטנמש בעפרון ומחק.
- בשאלון זה 15 דפים, כולל דף זה. ודא כי כל הדפים נמצאים.

בהצלחה!

ציון	ניקוד	
22	25	שאלה 1
18	25	שאלה 2
19	25	שאלה 3
23	25	שאלה 4
82	100	סה"כ



שאלה מס' 1 (25 נקודות)

א. לגבי טבלת הסמלים החיצוניים (ESD) עמכין האסמבלר, אילו מהטענות הבאות נכונות? תתכן יותר מתשובה אחת נכונה. הקף בעיגול את כל התשובות הנכונות.

$-\frac{1}{3}$

- i. הטבלה מכילה סמלים שערכם לא ידוע בזמן אסמבלי. ☒
- ii. הטבלה מכילה את כל הסמלים שערכם לא ידוע בזמן אסמבלי. ☐
- iii. הטבלה יכולה להיות ריקה. ☒
- iv. משתמשים בטבלה בתהליך הקישור (linking). ☒
- v. משתמשים בטבלה בתהליך הטעינה (loading). ☐

ב. לגבי הנחית האסמבלר model, אילו מהטענות הבאות נכונות? תתכן יותר מתשובה אחת נכונה. הקף בעיגול את כל התשובות הנכונות.

$-\frac{1}{2}$

- i. ההנחיה יכולה להופיע לכל היווהר פעם אחת בכל קובץ מקור. ☒
- ii. ההנחיה מגדירה ברירות מחדל לגבי סגמנטים וקבוצות (groups) בתכנית. ☒
- iii. ההנחיה נחוצה רק לצורך קישור עם תכניות בשפה עילית. ☐
- iv. הפרמטר use32 בהנחיה model. מאפשר את השימוש באוגרים הרב תכליתיים ברוחב 32 ביט. ☒
- v. האסמבלר יודיע על שגיאה אם ראה בתכנית הנחית code. לפני שראה הנחית model. ☐

ג. עבור כל אחד משלושת קטעי הקוד שלהלן, רשום את תוכנו של האוגר dx בגמר ביצוע הקטע. רשום את התשובה בבסיס 10.

<pre> xor dx,dx mov cx,8 push offset end3 loop3: jcxz ret3 push cx dec cx call loop3 pop cx add dx,cx ret3: ret end3: nop </pre>	<pre> .data string db '001000111001100010000' len dw len-string .code xor dx,dx mov di,@data mov es,di mov di,offset string mov cx,len mov al,'1' loop2: repnz scasb inc dx test cx,cx jnz loop2 </pre>	<pre> xor dx,dx mov cl,1 loop1: inc dx shl cl,cl jnz loop1 </pre>
dx = <u>36</u> ✓	dx = <u>8</u> ✓	dx = <u>3</u> ✓



המשך שאלה מס' 1 בדף הבא

שאלה מס' 1 (המשך)

ד. תרגם את הכתובת הלוגית (segment:offset) שלהלן לכתובת פיזית, כשהמעבד במצב real. רשום את התוצאה בבסיס 16.

8888h:0defh

8 9 6 6 5

ה. בצע את פעולות החיבור והחסור שלהלן בשיטת המשלים ל-2 ברוחב של 16 ביטים. כל המספרים נתונים בבסיס 16. רשום גם את התוצאות בבסיס 16. ציין את ערכי הדגלים CF ו-OF בגמר כל פעולה, כפי שהיו נקבעים על ידי ביצוע במעבד x86.

7163 + 1d6e 8FD1 ✓ CF=0 ✓ OF=1 ✓	7163 - 1d6e 5353 X CF=0 ✓ OF=0 ✓	7f34 + c3f3 4327 ✓ CF=1 ✓ OF=0 ✓	7f34 - c3f3 8B41 ✓ CF=1 ✓ OF=1 ✓
---	---	---	---

-1/4

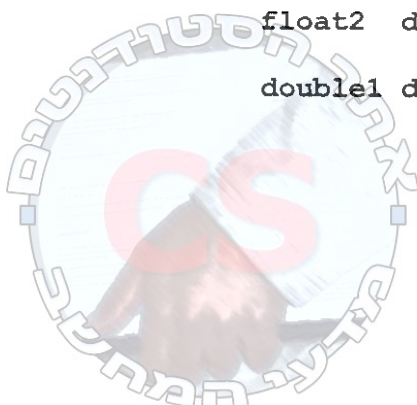
ו. תרגם את המספרים שבטבלה מבסיס 10 לייצוג סטנדרטי בשיטת הנקודה הצפה בבסיס 2. רשום את החזקה ללא bias (ראה דוגמא).

Decimal	Sign	Exponent	Mantissa
9.0	0	+3	1.001
-42.75	1 ✓	+5 ✓	1.0101011 ✓
1.25	0 ✓	+0 ✓	1.01 ✓

ז. להלן משתנים של תכנית אסמבלי, המכילים ערכים בשיטת הנקודה הצפה. תרגם את הערכים לבסיס 10 בייצוג ללא חזקה (ראה דוגמא). תזכורת: בשיטת הנקודה הצפה במחשב, נודה החזקה הוא מספר ללא סימן הכולל bias.

float1 dd 0c1180000h
float2 dd 0c22b0000h
double1 dq 3ff4000000000000h

-9.5
-42.75
+1.25



שאלה מס' 2 (25 נקודות)

הגדרה: "מיקום" (position) של איבר בתוך מערך הוא המרחק בבתים מתחילת המערך אל האיבר.

ברצוננו להשתמש במערך שתופס שטח גדול מ- 2^6 בתים. נקרא למערך כזה בשם "מערך גדול".
ברור כי במערך גדול קיימים איברים שהמיקום שלהם הוא מספר גדול מ- 2^{16} .

כידוע, במצב real, המעבד מאפשר גישה לזיכרון אך ורק לפי כתובת לוגית שההיסט בה קטן מ- 2^{16} .
מכיוון שהמיקום יכול להיות גדול מ- 2^{16} , אנו נזדקק לשיטה מיוחדת כדי לעבוד עם מערך גדול.

להלן השיטה בה נממש מערך גדול. המערך יתפרס על פני כמה סגמנטים (לא בהכרח רצופים בזיכרון).
גודלו של כל סגמנט של המערך יהיה 2^{16} בתינו בדיוק. נסמן כל סגמנט כזה במספר סידורי (החל מ-0).
המערך יתחיל בסגמנט שמספרו הסידורי 0, יתפרס על כל הסגמנט החל מהיסט 0, ואחר כך ימשיך
בסגמנט שמספרו הסידורי 1, וכן הלאה.

כתובות הסגמנטים שמרכיבים את המערך יוחזקו בטבלה, לה נקרא "מפת המערך".
הכניסה ה- i במפה ($i \geq 0$) מכילה את כתובת הסגמנט שמספרו הסידורי i .
(תזכורת: כתובת של סגמנט היא מספר ברוחב 16 ביטים, כרוחב אגרי הסגמנט).

דוגמא: להלן הקצאה של מערך גדול, המתפרס על שלשה סגמנטים, כלומר גודל המערך 3×2^{16} בתים.
המערך מורכב מהסגמנטים $seg0$, $seg1$, ו- $seg2$ (לפי סדר זה). מפת המערך נמצאת בסגמנט ה- $data$,
בכתובת $largeArrayMap$.

```

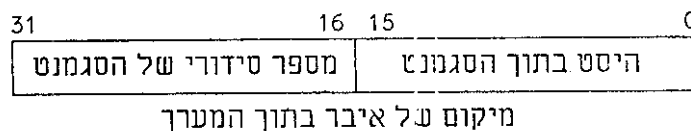
.model use16 small
.386
Seg0      segment
          db 10000h dup(?)
          ends

seg1      segment
          db 10000h dup(?)
          ends

seg2      segment
          db 10000h dup(?)
          ends

.data
largeArrayMap dw seg0
              dw seg1
              dw seg2
    
```

המיקום של איבר בתוך מערך גדול מיוצג כמספר בן 32 ביטים. בשיטה בה בחרנו לממש מערך גדול,
32 הביטים של המיקום נחלקים בטבעיות לשני עדות (ראה שרטוט להלן): 16 הביטים היותר משמעותיים
הם המספר הסידורי של הסגמנט בו נמצא האיבר, ו-16 הביטים הפחות משמעותיים הם ההיסט של האיבר
בתוך הסגמנט.



לדוגמא: האיבר במיקום $23456h$ במערך נמצא בסגמנט שמספרו הסידורי 2, בהיסט $3456h$ בתוך הסגמנט.
האיבר במיקום $9abch$ במערך נמצא בסגמנט שמספרו הסידורי 0, בהיסט $9abch$ בתוך הסגמנט.

המשך שאלה מס' 2 בדף הבא

שאלה מס 2 (המשך)

א. כתוב בשפת אסמבלי שגרה בשם `getSegment`, שכותרתה (בשפת C) נתונה להלן.

```
int getSegment(int *arrayMap, int segNum)
```

הפרמטר `arrayMap` הוא מצביע למפה של מערך גדול. המפה עצמה נמצאת בסגמנט ה-`data`. הפרמטר `segNum` הוא מספר סידורי של סגמנט במערך הגדול. השגרה מחזירה באוגר `ax` את כתובת הסגמנט מתוך מפת המערך. הנח כי הערך `segNum` לא חורג מגבולות המפה.

getsegment proc near

```
push bp
mov bp, sp
```

```
push ax
```

```
push bx
```

```
push di
```

```
mov ax, [bp+6]
```

```
mov di, ax
```

```
mov bx, [bp+4]
```

```
mov ax, [bx+di]
```

```
pop di
```

```
pop bx
```

```
pop bp
```

```
ret
```

```
endp
```

יש לקבל
2-1
כי הלכתי
הייתי מביא
לפי זה

ב. נתונה שגרה בשם `getBytes`, שכותרתה בשפת C היא:

```
char getBytes(int *arrayMap, long position)
```

הפרמטר `arrayMap` הוא מצביע למפה של מערך גדול. המפה עצמה נמצאת בסגמנט ה-`data`. הפרמטר `position` הוא מיקום בתוך המערך. השגרה מחזירה באוגר `al` את תוכן ה**בית** במיקום זה במערך. הנח כי המיקום אינו חורג מגבולות המערך.

להלן מימוש השגרה `getBytes` בשפת אסמבלי. השגרה קוראת לשגרה `getSegment` מסעיף א'.

```
1 .386
2 getByte proc
3 push bp
4 mov bp, sp
5 push bx
6 push es
7 push eax
8 mov eax, [bp+6]
9 mov bx, ax
10 shr eax, 16
11 push ax
12 push word ptr [bp+4]
13 call getSegment
14 add sp, 4
15 mov es, ax
16 pop eax
17 mov al, es: [bx]
18 pop es
19 pop bx
20 pop bp
21 ret
22 endp
```

i. מה מכיל האוגר `bx` בגמר ביצוע שורה 9?

Byte אליו אחרי

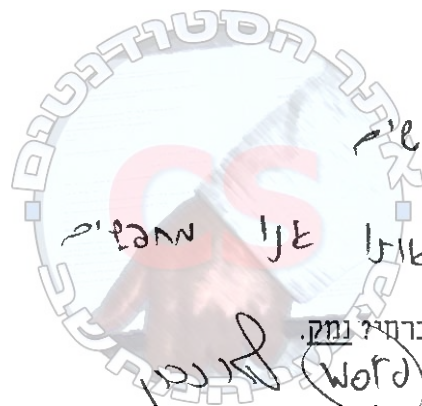
ii. מה מכיל האוגר `ax` בעת ביצוע שורה 11?

את מספר הסמל ב, נמצא ה-Byte

iii. האם ה-`casting` של האופרנד בשורה 12 (כדי שיצביע למילה) הינו הכרחי נח.

לא, כי אחסנית אפשר להכניס רק (word)

המשך שאלה מס' 2 בדף הבא



שאלה מס 2 (המשך)

ג. כתוב קטע קוד אשר מעתיק לאוגר al את הבית במיקום 12345h במערך הגדול שמוגדר בדף מס' 4 (מפת המערך היא largeArrayMap). חובה להשתמש בקריאה לשגרה getBytes מסעיף ב'.

~~12345h~~
 X dd 12345h
 push word ptr x
 push word ptr x+2
 push offset largeArrayMap
 call getByte
 add SP, 6

ד. הגדרה: "אינדקס" של איבר במערך חד מימדי הוא מספרו הסידורי של האיבר במערך (החל מ-0).

כשעובדים עם מערך חד מימדי שאיבריו בגודל מילה או יותר, נוח להשתמש באינדקס, ולא במיקום, וזאת מכיוון שהאינדקס אינו תלוי בגודל אברי המערך. כמובן שניתן לחשב את המיקום, בהינתן האינדקס וגודל האיברים.

במערך גדול, יתכן ויש יותר מאשר 2^{16} איברים. לפיכך, האינדקס של איבר במערך גדול ייוצג תמיד כמספר בן 32 ביטים.

כתוב בשפת אסמבלי שגרה בשם getWord, שכותרתה (בשפת C) נתונה להלן.

```
int getWord(int *arrayMap, long index)
```

הפרמטר arrayMap הוא מצביע למפה של מערך גדול. המפה עצמה נמצאת בסגמנט ה-data. המערך הוא חד מימדי ואיבריו בגודל מילה. הפרמטר index הוא אינדקס של איבר (מילה) במערך. השגרה מחזירה באוגר ax את תוכן האיבר באינדקס זה במערך. הנח כי האינדקס אינו חורג מגבולות המערך.

מותר להשתמש באוגרים הרב תכליתיים המורחבים (ברוחב 32 ביטים).
רמז: מומלץ לקחת את השגרה getBytes מסעיף ב' ולבצע בה שנויים כנדרש.

```

getWord proc
    push BP
    mov BP, SP
    push ESI
    push EAX
    mov EAX, [BP+6]
    mov BX, AX
    shr EAX, 16
    push AX
    push word ptr [BP+4]
    call getSegment
    add SP, 4
    mov ES, AX
    mov AX, BX
    mov BX, 2
    mul BX
    mov BX, AX
    pop EAX
    
```

```

mov AX, ES:[BX]
pop ESI
pop BX
pop BP
ret
endp
    
```

המשך שאלה מס' 2 בדף הבא

אל תשכח
 ל-2 יש קריאה
 ו-2 בקריאה, אל תשכח
 כ. גם אם תשכח
 (אם תשכח)

שאלה מס 2 (המשך)

ה. כתוב בשפת אסמבלי שגרה בשם `copySegment`, שכותרתה (בשפת C) נתונה להלן.

```
void copySegment(int targetSegment, int sourceSegment)
```

שני הפרמטרים של השגרה הם כתובות של סגמנטים. שני הסגמנטים בגודל 2^{16} בתים. השגרה מעתיקה את תוכן הסגמנט `sourceSegment` אל הסגמנט `targetSegment`. חובה להשתמש בהעתקה מהירה של מחרוזות, על ידי אחת הפקודות `movsb`, `movsw` או `movsd`.

copy segment proc real

push bp
mov bp, sp

pusha

mov ax, [bp+6]

mov ds, ax

xor si, si

mov ax, [bp+4]

mov es, ax

xor di, di

mov cx, 055555h

rep movsb

popa

pop bp

ret

endp

-1/2/6

*יש להחזיר
312 ק"מ
את ה-1*

ה. כתוב בשפת אסמבלי שגרה בשם `copyLargeArray`, שכותרתה (בשפת C) נתונה להלן.

```
void copyLargeArray(int *targetMap, int *sourceMap, int numSegs)
```

שני הפרמטרים הראשונים הם מצביעים למפות של מערכים גדולים. המפות עצמן נמצאות בסגמנט ה-`data`. שני המערכים בגודל זהה. הפרמטר `numSegs` הוא מספר הסגמנטים בכל אחד מהמערכים. השגרה מעתיקה את תוכן המערך שמוגדר על ידי המפה `sourceMap` אל המערך שמוגדר על ידי המפה `targetMap`. חובה להשתמש בקריאה לשגרה `copySegment` מסעיף ה'.

copy large array proc real

push bp

mov sp, bp

pusha

mov cx, [bp+8]

mov dx, [bp+6]

mov bx, [bp+4]

loop ||

popa

pop bp

ret

endp

||:

push dx

push bx

call copySegment

add sp, 4

add dx, 2

add bx, 2

-2/5

*הסגמנטים
אלו אינם סגמנטים
אלו סגמנטים
לדוגמה -
כדי להחזיר
את ה-1*

י. כידוע, כאשר המעבד נמצא במצב `real`, מרחב הכתובות הפיזיות הוא 2^{20} בתים כמה סגמנטים לכל היותר יכולים להיות במערך גדול במצב `real`?

*2.2.2.2
2.2.2.2*

שאלה מס' 3 (25 נקודות)

א. להלן הגדרת המאקרו `jcond`. הפרמטר `dest` זהה לאופרנד של פקודת המכוונה `jmp`. כל אחד מארבעת הפרמטרים הנותרים נושא שם של אחד מדגלי התנאי באוגר הדגלים. כל פרמטר כזה הוא הקבוע 0 או 1. המאקרו מבצע הסתעפות לכתובת `dest` אם ערכי דגלי התנאי באוגר הדגלים זהים בהתאמה לערכי הפרמטרים. בקריאה למאקרו מותר שלא להעביר אחד או יותר מארבעת הפרמטרים האחרונים. הדגלים המתאימים לפרמטרים החסרים לא ישתתפו בבדיקת התנאי להסתעפות.

```
jcond macro dest,zf,sf,cf,of
    local nojump
    ifnb <&zf>
        if zf eq 0
            jz nojump
        else
            jnz nojump
        endif
    endif
    ifnb <&sf>
        if sf eq 0
            js nojump
        else
            jns nojump
        endif
    endif
    ifnb <&cf>
        if cf eq 0
            jc nojump
        else
            jnc nojump
        endif
    endif
    ifnb <&of>
        if of eq 0
            jo nojump
        else
            jno nojump
        endif
    endif
    jmp dest
nojump: nop
endm
```

להלן שלש דוגמאות של קריאות למאקרו `jcond`. עבור כל דוגמא, רשום את הקוד בשפת אסמבלי שמתקבל מפרישת הקריאה למאקרו. יש לרשום רק שורות שיצורות קוד מכונה (אין לרשום את ההנחיות לאסמבלי מותנה, וכד'). השתמש בערך 00000?? עבור פרישת התווית המקומית `nojump`.

<code>jcond t,0,0,1,1</code>	<code>jcond t,,,1,1</code>	<code>jcond t</code>
<code>JZ 00000??</code> <code>JS 00000??</code> <code>JNC 00000??</code> <code>JNO 00000??</code> <code>JMP t</code> <code>00000??:nop</code>	<code>JHC 00000??</code> <code>JHD 00000??</code> <code>JMP t</code> <code>00000??:nop</code>	<code>JMP t</code> <code>00000??:nop</code>

המשך שאלה מס' 3 בדף הבא

שאלה מס' 3 (המשך)

- i. הגדר מאקרו בשם `jbe`, שמקבל פרמטר אחד הזהה לאופרנד של פקודת המכונה `jmp`. המאקרו `jbe` מבצע הסתעפות מותנית באותו התנאי כמו פקודת המכונה `jbe`. המאקרו אינו משנה את אוגר הדגלים. חובה להשתמש בקריאות למאקרו `jcond` מסעיף א'. אסור להשתמש ישירות בפקודות המכונה להסתעפות מותנית.

Handwritten notes:
`Jbe Macro dest`
`Jcond dest, 1, 1, 1`
`endm`
 $1\frac{1}{2}/2$
 צורה הזן ה-10
 $2^4 = 16$
 16 בייט
 16 בייט
 16 בייט
 16 בייט

- ii. ציין לפחות יתרון אחד וחסרון אחד של המאקרו `jbe` בהשוואה לפקודת המכונה `jbe`.

Handwritten notes:
 חסרון - המאקרו יכריז קוד יוני ארוך
 יתרון - מקנה אופרנד `OPCODE` ל-`jbe`
 $1\frac{1}{2}/1$

- ג. הגדר מאקרו בשם `joc`, שמקבל פרמטר אחד הזהה לאופרנד של פקודת המכונה `jmp`. המאקרו `joc` מבצע הסתעפות מותנית אם מתקיים התנאי `OF=CF`. המאקרו אינו משנה את אוגר הדגלים. חובה להשתמש בקריאות למאקרו `jcond` מסעיף א'. אסור להשתמש ישירות בפקודות המכונה להסתעפות מותנית.

Handwritten code:
`joc macro dest`
`Jcond dest, 1, 1, 1`
`Jcond dest, 1, 0, 0`
`endm`

- ד. הגדר שלשה מאקרוים בשם `setbit`, `clrbit`, `combit`, שכותרתיים נטונות להלן. הפרמטר `src` הוא מקום בזיכרון בכל שיטת מעון, או אוגר רב תכליתי. רוחבו של `src` הוא 8, 16 או 32 ביטים. הפרמטר `n` הוא מספר קבוע המציין את אחד הביטים של `src`. לפי הסדר המקובל (ביט מס' 0 הוא הביט הפחות משמעותי). המאקרו `setbit` מדליק את הביט ה-`n` של `src`, המאקרו `clrbit` מכבה את הביט, ואילו המאקרו `combit` הופך את ערכו של הביט. אין שינוי בערכי הביטים האחרים של `src`. מותר למאקרוים לגרום לשינוי באוגר הדגלים. הנח כי הערך `n` אינו חורג מהרוחב של `src`. דמיון: מומלץ להשתמש בפקודות הסיבוב.

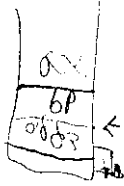
<pre>combit macro src, n local aa, bb test src, h Test src, 1 JE aa rcl src, h clrbit src, n jmp bb aa: rcl src, h setbit src, n bb: endm</pre>	<pre>clrbit macro src, n local aa test src, h Test src, 1 JE aa and src, 1 aa: rcl src, h endm</pre>	<pre>setbit macro src, n local aa test src, h or src, 1 rcl src, h endm</pre>
---	--	---

המשך שאלה מס' 3

-1/6

שאלה מס' 3 (המשך)

ה. להלן הגדרת המאקרו `decr`. המאקרו מקבל פרמטר אחד, שהוא אוגר רב תכליתי. המאקרו מקטין ב-1 את תוכן האוגר, בדומה לפקודת המכונה `dec`, אך בהבדל אחד: המאקרו אינו משנה את הדגלים.



```

1  decr macro op
2      pushf
3      dec op
4      ifidni <op>,<sp>
5          push bp
6          mov bp,sp
7          push ax
8          mov ax,[bp+3]
9          mov [bp+2],ax
10         pop ax
11         pop bp
12     endif
13     popf
14 endm

```

i. מדוע נחוצות השורות 4-12? במקרה שהאופרנד הוא `sp` צריך מקרה מיוחד לאגור בו

ii. האם המאקרו `decr` יעבוד נכון גם עבור פרמטר שהוא מקום בזיכרון נמוך. כן, הקופצה `dec` משלמת את הפרמטרים והיזכרון

ו. הגדר מאקרו בשם `incr`, המקבל פרמטר אוגר שהוא אוגר רב תכליתי. המאקרו מגדיל ב-1 את תוכן האוגר, בדומה לפקודת המכונה `inc`, אך מבלי לשנות את הדגלים.

`incr macro op`

`pushf`

`inc op`

`ifidni <op>,<sp>`

`push bp`

`mov bp,sp`

`push ax`

`mov ax,[bp+3]`

`mov [bp+2],ax`

`pop ax`

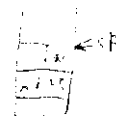
`pop bp`

`inc sp`

`popf`

`endm`

המשך שאלה מס' 3 בדף הבא



לפי הסבר הקורס
שה חורא
הכנאים המאקרו

-1/3

A hand-drawn diagram of a container, possibly a beaker or test tube, with a label 'bp' at the top and 'ax' below it. The container is partially filled with a liquid, indicated by a horizontal line.

מימוש תקין		מימוש פגום	
1	pushb macro op	1	pushb macro op
2	decr sp	2	decr sp
3	push ax	3	push ax
4	push bp	4	push bp
5	mov al,op	5	mov bp,sp
6	mov bp,sp	6	mov al,op
7	mov [bp+4],al	7	mov [bp+4],al
8	pop bp	8	pop bp
9	pop ax	9	pop bp ax
10	endm	10	endm

27
a x

push b mact0 bje ptr [bp]

[illegible]

יש לבצע את כל החישובים בזמן אסמבלי, תוך שימוש במבני אסמבלי מותנה מתאימים. קוד המכונה שנוצר מכיל אך ורק את הטבלה עצמה (נו"כ 512 בתים). הערה: מומלץ להשתמש במבני **irp** או **irpc**.

~~a mach6
x=0
while x lt 16
db '0', x+30
x=x+1
endm
while x lt 16
db 'x', x+30
x=x+1
endm
while x lt 16
db '2', x+30
x=x+1
endm~~

macro DO
y=0, x=0

while y 17/16

White x 17 16

$(26 \times 80, x+30)$
pho m $x = x+1$

$$y = y + 1$$

Enj m

$$-1\frac{1}{2} / 3$$

קורב כל, בל. 30h. 180 סי. זקלם
חול למה, עבור 1500
http://cs.haifa.ac.il/students/
תת הסטודנטים - החוג למדעי המחשב, אוניברסיטת חיפה

שאלה מס' 4 (25 נקודות)

כידוע, פסיקה מס' 1, הידועה בשם פסיקת single-step, נוצרת בגמר מחזור הפקודה בתנאי שהדגל TF באוגר הדגלים היה דלוק בתחילת אותו מחזור פקודה (מלבד, כידוע, מחזור הפקודה int, וכמה מקרים נוספים). לצערנו, לא קיימת פקודת מכונה להדלקה או כיבוי הדגל TF. ברצוננו לספק אמצעי נוח לשינוי הדגל TF, וזאת כדי לאפשר לתכנית המשתמש לשלוט במנגנון ה-single-step. האמצעי שנבחר הוא הפסיקה מס' 0x6f. זוהי פסיקה פנויה, ואנו נגדיר את השרות שלה כדלקמן.

שרות הפסיקה 0x6f יפעל כמתג הדלקה/כבוי של הדגל TF. אם הדגל TF באוגר הדגלים היה כבוי בעת שנוצרה הפסיקה 0x6f, הדגל יודלק בעת החזרה משרות הפסיקה. אחרת, הדגל יכובה בעת החזרה מהשרות. שרות הפסיקה 0x6f מקבל כפרמטר את האוגר dx. כאשר השרות מדליק את הדגל TF, ערך האוגר dx קובע כמה פעמים לכל היותר תיווצר פסיקה מס' 1. לאחר שנוצרה כמות זו של פסיקות מס' 1, הדגל TF יכובה על ידי שרות הפסיקה 1 עצמו.

כמקרה מיוחד, אם האוגר dx מכיל 0, שרות הפסיקה 0x6f יאפשר לפסיקה מס' 1 להיווצר מספר בלתי מוגבל של פעמים, עד להפעלה הבאה של הפסיקה 0x6f, אשר תדאג לכבות את הדגל TF.

כאשר שרות הפסיקה 0x6f מכבה את הדגל TF, האוגר dx אינו משמש כפרמטר.

להלן תכנית המיישמת את שרותי הפסיקות מס' 0x6f ומס' 1, כמוגדר לעיל. שים לב: התוכנית עוברת אסמבלי וקישור על ידי הפקודה BCC (הקומפילר של שפת C).

1	.stack 100h	38	_main: mov ax, 3501h
2	.data	39	int 21h
3	oldInt1 dd ?	40	mov word ptr oldInt1, bx
4	oldIntF6 dd ?	41	mov word ptr oldInt1+2, es
5	trapCount dw 0	42	mov ax, 35f6h
6	formStr db "%04x", 13, 10, 0	43	int 21h
7	.186	44	mov word ptr oldIntF6, bx
8	.code	45	mov word ptr oldIntF6+2, es
9	public _main	46	push ds
10	extrn _printf: near	47	mov ax, cs
11	newIntF6: push bp	48	mov ds, ax
12	mov bp, sp	49	mov dx, offset newInt1
13	test word ptr [bp+6], 100h	50	mov ax, 2501h
14	jnz stopInt1	51	int 21h
15	startInt1: or word ptr [bp+6], 100h	52	mov dx, offset newIntF6
16	mov trapCount, dx	53	mov ax, 25f6h
17	pop bp	54	int 21h
18	iret	55	pop ds
19	stopInt1: and word ptr [bp+6], 0feffh	56	xor ax, ax
20	mov trapCount, 0	57	mov cx, 100
21	pop bp	58	mov dx, 8
22	iret	59	int 0f6h
23	newInt1: push bp	60	lp1: inc ax
24	mov bp, sp	61	loop lp1
25	pusha	62	xor ax, ax
26	push word ptr [bp+2]	63	mov cx, 100
27	push offset formStr	64	mov dx, 0
28	call _printf	65	int 0f6h
29	add sp, 4	66	lp2: inc ax
30	cmp trapCount, 0	67	loop lp2
31	je exitInt1	68	int 0f6h
32	dec trapCount	69	xor ax, ax
33	jnz exitInt1	70	push ds
34	and word ptr [bp+6], 0feffh	71	mov ax, 2501h
35	exitInt1: popa	72	lds dx, oldInt1
36	pop bp	73	int 21h
37	iret	74	lds dx, oldIntF6
		75	mov ax, 25f6h
		76	int 21h
		77	pop ds
		78	mov ah, 4ch
		79	int 21h
		80	ret
		81	end

המשך שאלה מס' 4 בדף הבא

שאלה מס' 4 (המשך)

א. השאלות x-i שלהלן מתייחסות להבנת התכנית שבדף הקודם. מומלץ לעקוב אחרי התכנית לפי סדר השאלות, ולעבור על כל השאלות מראש, לפני כתיבת התשובות.

i. מה עושות שורות 41-38? מה עושות שורות 45-42?

38-41 - מלא את הקטע הבסיסי 01 הישן
42-45 - מלא את הקטע הבסיסי f6 הישן

ii. מה עושות שורות 51-47? מה עושות שורות 54-52?

47-51 - מאתחיל את הבסיס 01 הישן
52-54 - מאתחיל את הבסיס f6 הישן

iii. מה עושות שורות 14-11? באיזה מקרה מתבצעת ההסתעפות בשורה 14?

11-14 - קוצר את הבסיס TF

מורה 14 מתקדם האם TF קוצר

iv. מה עושות שורות 18-15? באיזה מצב נמצא הדגל TF באוגר הדגלים אחרי בצוע שורה 18?

הדגל הוא - הוריות נמצאות את TF ומעבירות למינה את דגל
ה TF יהיה נכון אחר קוצר מורה 18

v. מה עושות שורות 22-19? באיזה מצב נמצא הדגל TF באוגר הדגלים אחרי בצוע שורה 22?

הדגל הוא - האם TF נמצא את המינה

הבסיס TF יהיה כקוד קוצר מורה 22

vi. מהו מספר השורה שבגמר מחזור הפקודה שלה נוצרת בפעם הראשונה בתכנית פסיקת single-Step מהו תוכנו של המשתנה trapCount כשנוצרת הפסיקה? כיצד נקבע תוכן זה? הסבר.

מורה 60 תוכן trapCount הוא 8 הוא מוצר 8

vii. מה מדפיסה השגרה printf בקריאה בשורה 28? תן הסבר כללי, אין צורך לרשום את הפלט.

המשתנה מצביע את תוכן המלא ממשלה - BP - 5

viii. מה עושה שורה 34? מהו תוכנו של המשתנה trapCount בעת בצוע שורה זו?

אם שורה 34 אכן מתבצעת, מה היה תוכנו של trapCount בעת הכניסה לשגרת הפסיקה מס' 1?

34 - מכבה את TF - trapCount מכיל 0

אם מורה 13 קוצרה, תוכן היה 1

ix. מה היה תוכנו של trapCount בעת הכניסה לשגרת הפסיקה מס' 1 אם שורה 32 אינה מתבצעת?

מה היה תוכנו של trapCount בעת היציאה משגרת הפסיקה במקרה זה?

תוכן הוא 0

תוכן יהיה 1 אם מורה 14

x. הסבר בקצרה את תפקידו של המשתנה trapCount בשגרת השרות של פסיקה מס' 1.

trapCount מניח את מספר הפעמים בהם קוצרה פסיקה
1 כאשר הוא מניח 0 הוא מכבה את ה TF יוצא
אם היה 0 על הבסיס תהיה תמיז - לא מעביר TF

המשך שאלה מס' 4 בדף הבא

שאלה מס' 4 (המשך)

ב. כמה פעמים תיווצר פסיקת single-step במשך ביצוע קטע הקוד בשורות 62-57? מהו תכנו של האוגר ax בעת שנוצרת פסיקת single-step האחרונה בקטע קוד זה? נמק.

כמה פעמים

ax=4 קודם לא היה שם וכל פעם בלולאה יש
שתי עקבות.

ג. כמה פעמים תיווצר פסיקת single-step במשך ביצוע קטע הקוד בשורות 69-63? מהו תכנו של האוגר ax בעת שנוצרת פסיקת single-step האחרונה בקטע קוד זה? נמק. תזכורת/רמז: מחזור הפקודה int 0f6h אינו יוצר פסיקת single-step.

200 במחיר כי מאלף ax=0 מה שניתן לפסקה לעיבוד מספר 200
בזמן מילוי של פעמים
בשלב ביצע מילוי של ax=100

ד. מה תהיינה תשובותיך לסעיף ג' אם נחליף את שורה 64 בשורה: mov dx,8 ?

כאן קודם ל' וצריך להוסיף 310 פסקה אחת
כאן הפסקה שלק את המלש ב שורה 64
ואכן ax=0 יתן את פסקה - עכ"ל 201
ax יהיה עכ"ל שורה 0-8 כי הם מילוי של פסקה אחת
? mov trapCount,1 מה תהיינה תשובותיך לסעיף ג' אם נחליף את שורה 68 בשורה: הערה: שורה 64 היא השורה המקורית, ללא השנוי מסעיף ד'.

ax "100" 310 פסקה אחת 201
אבל הפסקה קודם 100 פסקה אחת 201

ה. האם אפשר לוותר על שורה 20? נמק.

כן, הפסקה של trapCount
לא משנה 180 פסקה אחת (כדי)



המשך שאלה מס' 4 בדף הבא

שאלה מס' 4 (המשך)

ז. הקוד של שגרת הפסיקה מס' 1 (שורות 23-37) אינו משנה אף אוגר רב תכליתי מלבד bp (וכמובן sp). האם אפשר לוותר על שמירת האוגרים הריב תכליתיים בשורה 25 ושחזורם בשורה 35? נמק.

האם אפשר לוותר על שמירת האוגרים הריב תכליתיים בשורה 25 ושחזורם בשורה 35? נמק.
 אכן, שכן יש לנו קריאה לפונקציה אחרת
 אכן, שכן יש לנו קריאה לפונקציה אחרת
 אכן, שכן יש לנו קריאה לפונקציה אחרת

ח. שני שגרות הפסיקה משתמשות במשתנים וסגמנט ה-data, ולמרות זאת אין בתכנית כולה אף שורה שמאתחלת את האוגר ds לכתובת סגמנט ה-data. האם יש כאן בעיה? נמק.

כן, קראתם שגרירת הוידאו באשר כנים למספרים
 שני הוידאוים מסומנים על ידי
 שני הוידאוים מסומנים על ידי
 שני הוידאוים מסומנים על ידי

ט. מדוע נחוצה שורה 29?

אכן, שכן יש לנו קריאה לפונקציה אחרת

אכן, שכן יש לנו קריאה לפונקציה אחרת
 כיצד ניתן להפעיל את פסיקה מס' 1 כאשר הדגל TF באוגר הדגלים כבוי?
 אכן, שכן יש לנו קריאה לפונקציה אחרת

היא תהיה
 BCC

היא תהיה
 iht 1h

