**ארכיטקטורת מחשב – תרגיל בית 03**

**אסמבלי – זיכרון, תנאים ולולאות**

**כללי:**

נושאי התרגול:

* עבודה עם זיכרון
* כתיבת לוגיקה פשוטה באסמבלי

לאורך כל הסמסטר העבודה היא אישית - אסור לשבת לעבוד ביחד. עם זאת, מותר ואפילו מומלץ להתייעץ אחד עם השני.

במידה ואתם מתקשים נסו להעזר בגוגל ובמידה ועדיין אתם מתקשים פנו למדריך!

**הוראות להגשת התרגיל:**

1. עבור כל התכניות שתכתבו באסמבלי, אנא השתמשו בתבנית הבאה:

org 100h

[your code here]

mov ah, 0

int 16h

ret

1. יש לשמור את הקבצים עם סיומת asm, כאשר שם הקובץ כולל את מספר השאלה. לדוגמא, תוכנית עבור שאלה מס' 1, ישמר בקובץ שנקרא 1.asm.
2. לאחר שכתבתם את הקוד, הריצו את תכניתכם ועקבו אחר פעולתה באמצעות כפתור emulate. וודאו כי תכניתכם פועלת כראוי ומבצעת את הנדרש.
3. יש להגיש קובץ zip (בלבד!) המכיל את כל התוכניות. יש להקפיד על השם של הקובץ לפי התבנית הבאה:

03\_israel\_israeli.zip (כאשר 03 מציין את מס' העבודה, וישראל ישראלי הוא שם פרטי ומשפחה).

***המשך בעמוד הבא***

## שאלה 1

פתחו את הקובץ ***addressing.asm*** באמצעות emu8086. לפניכם שאלה עם חמישה חלקים המסומנים A,B,C,D,E. עקבו אחר ההוראות בכל חלק:

* 1. חלק A – השתמשו בפעולות אריתמטיות על המשתנים num1,num2,num3 כדי להגיע לתוצאה הרצוייה 23. אין להשתמש במספרים נוספים (immediates).
  2. חלק B – הדפיסו את התו השלישי במערך התווים chrs. אל תשתמשו ישירות ב-chrs, אלא "הגיעו" למשתנה מתוך המצביע chrs\_ptr.
  3. חלקC – הדפיסו את הערך 15 ע"י שימוש באוגרים הנתונים ax=10, bx=4 ובפקודה lea

(זהירות: חישוב הכתובות במעבד 8086 "רגיש" לאוגרים בהם משתמשים. כלומר אי אפשר לעשות למשל [bx+ax]. קרא באילו אוגרים מותר להשתמש).

* 1. חלקD – שנו את הגדרת המשתנים static\_num1\_ptr ו-static\_num1\_ptr\_ptr שבראש התוכנית, כדי שהקוד הקיים של חלק D (ללא שינויים) ידפיס את הערך הנכון 7.
  2. חלקE – עליכם לכתוב מספר שורות קוד שישימו ערכים נכונים למשתנים dynamic\_num1\_ptr ו-dynamic\_num1\_ptr\_ptr, כך שהקוד הקיים של חלק E ידפיס נכון את הערך 7. אסור לשנות את הגדרת משתנים אלה ישירות, אלא רק לשנות את הערכים שלהם בזמן ריצה.

## שאלה 2

כתבו תכנית אשר בודקת אם ערך מסוים שיוצב באוגר **ax** מתחלק ב-7 ללא שארית.

בשני המקרים (מתחלק \ לא מתחלק), התוכנית תדפיס הודעה מתאימה למשתמש.

## שאלה 3

כתבו תכנית אשר תדפיס את 20 האיברים הראשונים בסדרת פיבונאצ'י לפי הסדר.

## שאלה 4

קראו בספר על פעולות כפל וחילוק. כתבו תכנית אשר מחשבת (ומדפיסה) עצרת של מספר מסוים שערכו נמצא באוגר **ax**.

***המשך בעמוד הבא***

## שאלה 5

כתבו תכנית אשר מחשבת ומדפיסה את כמות הביטים הדולקים באוגר **ax**.

רמז: קראו מה עושה בדיוק הפעולה shl\shr וחשבו איך באמצעותה ובעזרת אחד הדגלים ניתן לפתור את השאלה.

## שאלה 6 - בונוס

אתם וודאי מכירים את מושג ה-cast משפת C. קראו באינטרנט על פקודות **movzx** ו-**movsx**.

הערה: שימו לב שהאמולטור emu8086 לא תומך בפקודות אלו, ולכן לא תוכלו להשתמש בהן.

הנחיות:

1. הניחו שהטיפוס short הוא מספר שלם בגודל 16 סיביות (2 בתים). כתבו מימוש עבור כל אחת משתי הפעולות הבאות:

הערה: בדקו שהקוד שלכם עובד. לדוגמא, אם הערך שהיה שמור בתוך אוגר bl היה 5- (כאשר הוא היה מיוצג בעזרת 8 סיביות), וודאו שערכו נשאר זהה לאחר ההשמה לאוגר ברוחב 16 סיביות.

1. תנו דוגמא לערך אחד של bl, עבורו הערך המתקבל ב-ax אחרי שתי הפקודות יהיה זהה זה לזה.
2. תנו דוגמא לערך אחר של bl, עבורו הערך המתקבל ב-ax אחרי שתי הפקודות יהיה שונה זה מזה.

## שאלה 7 - בונוס

לפניכם קטע קוד שמשתמש בפקודת switch:

switch (si)

{

case 1:

printf( “Number One” );

break;

case 2:

printf( “Number Two” );

break;

case 3:

printf( “Number Three” );

break;

case 4:

printf( “Number Four” );

break;

}

***המשך בעמוד הבא***

בשאלה זו תממשו קטע קוד זה באסמבלי ללא שימוש בתנאים (כלומר ללא שימוש בפקודות: ja, jl, jge ודומיהן), אלא ע"י שימוש בפקודת jmp ללא תנאי, ובמבנה מסוג branch table.

1. עיין בערך הויקיפדיה <http://en.wikipedia.org/wiki/Branch_table>
2. ממש את קטע הקוד הנ"ל באסמבלי תוך שימוש בשיטה זו.

תזכורת: בשביל ביצוע הדפסות:

* 1. שים את הקובץ magshimim.inc באותה התיקייה כמו הקוד שלך
  2. הוסף בסוף הקובץ את השורה include magshimim.inc
  3. בשביל להדפיס, השתמש בשורה PRINTN “Number One”

הערה: אתה יכול להניח שאין צורך לטפל במקרה בו si מכיל מספר מחוץ לטווח 1-4.

רמז א

הרמז מוסתר מאחורי המלבן. הזז אותו כדי להציץ.

השתמש בקוד שמבצע קפיצה בעזרת מצביע:

mov bx, address

jmp bx

address:

dw offset foo

foo:

PRINTN “Foo”



**בהצלחה!**

הזז אותי בשביל להציץ ברמז ☺

רמז ב

הרמז מוסתר מאחורי המלבן. הזז אותו כדי להציץ.

צור טבלה של כתובות ל-case-ים השונים. כדי לבחור בערך הנכון בטבלה השתמש בקוד כמו:

//assume the register si contains the “switch variable”

mov bx,table\_start

sub si,1 //table starts from zero

add si,si //each entry in the table is 2 bytes long

mov bx,[bx+si]