ארגון ותכנות המחשב

תרגיל בית 1 (רטוב)

המתרגל האחראי על התרגיל: עידו סופר.

הנחיות:

- שאלות על התרגיל ב- **Piazza** בלבד.
- ההגשה **בזוגות**.
- על כל יום איחור או חלק ממנו, שאינו באישור מראש, יורדו 5 נקודות.
    - ניתן לאחר ב-3 ימים לכל היותר.
    - הגשות באיחור יתבצעו דרך אתר הקורס.
- יש להגיש את התרגיל בקובץ zip לפי ההוראות בעמוד הבא. אי עמידה בהוראות אלו תעלה לכם בנקודות יקרות.
- תיקונים לתרגיל, אם יהיו, יופיעו ממורקרים.
- כל תרגיל מופיע בעמוד נפרד.

מבוא

הוראות כלליות

- התרגיל וכל ההוראות כתובים בלשון זכר/נקבה באופן אקראי אך פונים לגברים ונשים באופן שווה.
- ישנם 5 סעיפים, הפתרון של כל אחד צריך להופיע בקובץ נפרד ex1.asm, ex2.asm וכו' המצורפים לכם להשלמה. ההגשה שלכם צריכה להיות קובץ zip אחד שמכיל את חמשת הקבצים שקיבלתם, בלי שהורדתם מהם כלום וכאשר הוספתם להם רק פקודות אסמבלי ולא שום דבר אחר.

טסטים

- מסופק לכם קובץ בדיקה לתרגיל 1 והוראות כיצד להשתמש בו. הטסט מכסה מקרה בסיסי ומסופק לכם כדי שתדעו כיצד אנו מצפים לקבל את הפתרון (בכל אחד מחמשת התרגילים) וכיצד אנו מתכוונים לבדוק אותם.
- מומלץ ביותר לבצע טסטים נוספים בעצמכם לכל חמשת התרגילים, על בסיס מבנה הטסט הזה.

חוקים לכתיבת הקוד

- בקוד שאתם מגישים אין לשנות שום ערך בזיכרון מלבד אלו שנאמר לכם מפורשות.
- עליכם לבצע כל חישוב אפשרי ברגיסטרים. ביצוע חישובים על ערכים בזכרון הוא לא יעיל.
- בכל סעיף הניחו כי בתוויות הפלט הוקצה מספיק מקום.
- על כל תווית שתשמשו בה לצורך מימוש להסתיים בארבעת התוים HW1_. מה שמופיע לפני מומלץ כמובן שיהיה קשור למטרת התווית לצורך נוחות שלכם ולצורך קריאות של הקוד.

דוגמה:

```
loop: #this is a bad label name
loop_HW1: #this is a good label name
```

זכרו - אי עמידה בכל אחת מההוראות הנ"ל עלולה לגרור הורדת ניקוד ולהשפיע על זכותכם לערעור במקרים מסוימים.

חוקים לשאלות בפיאצה (או – למה לא עונים לי ?)

- נא לקרוא את כל המסמך לפני שבכלל נכנסים לפיאצה, בטח ובטח לפני ששואלים משהו.
- נא לקרוא את השאלות ששאלו לפניכם. אתם יכולים רק להרוויח מכך וסביר שזה גם יקרה.
- כל מקרה קצה שלא מופיע מפורשות בהוראות הסעיף אפשרי. התרגיל נכתב בקפידה ותחת ווידוא שלכל מקרה קצה כנ"ל יש תוצאה מצופה שניתנת להבנה מהוראות התרגיל ומקרי קצה הם כמובן, חלק מהקורס וחלק מהחיים כמתכנת. כמובן שהסגל אנושי ויכול לפספס משהו וכל שאלה בפיאצה תקרא ובמידה ועומדת בכל הסעיפים בחלק זה של ההוראות, גם תענה.
- אנו לא מדבגים לכם. זה חלק מהקורס כמו בכל הקורסים בפקולטה.
- שאלות פרטניות ובפרט כאלה הכוללות תמונות קוד, נא לכתוב כ-private, שלא חשופה לשאר הכיתה.

בדיקות וטסטים

כאמור, כל חמשת התרגילים יבדקו בצורה דומה. לכן, עקבו אחר ההוראות שיתוארו להלן, כאשר תרצו לבדוק את הקוד שלכם לפני ההגשה. חבל שההגשה שלכם לא תהיה לפי הפורמט ותצטרכו להתעסק עם ערעורים ולאבד נקודות סתם.

בכל תרגיל, תקבלו קובץ asm המכיל text section בלבד. עליכם להשלים את הקוד שם, אך לא להוסיף sections נוספים לקובץ בעת ההגשה (<u>ההגשה חייבת להכיל **section text** בלבד. בפרט **לא** להכיל את **section data**</u>).

אז איך בכל זאת תוכלו לבדוק את התרגיל שלכם? זה פשוט. לתרגיל 1 מצורף טסט בודד בתיקייה tests. הבדיקה היא בעזרת הקובץ run_test.sh.

שימו לב שכל הטסטים על קבצי הקוד שלכם ירוצו עם timeout (כפי שגם מופיע ב-`run_test.sh`) ולכן כתבו אותם ביעילות. קוד שלא ייכתב ביעילות ולא יסיים את ריצתו על טסט מסוים עד ה-timeout, ייחשב כקוד שלא עמד בדרישות הטסט. בפרט, הקוד ייבדק באותו מגבלת זמן שמופיעה בטסטים (ועל מחשב עם יכולות עיבוד נמוכות מהממוצע), כך שאתם תראו שאין באמת צורך לאלגוריתמים פורצי דרך כדי לעמוד בהם, רק לעבוד בגבולות ההיגיון.

הריצו את הקובץ run_test.sh באופן הבא:

```
./run_test.sh <path to asm file> <path to test file>
```

<u>**דוגמה**</u>: עבור התרגיל **הראשון** והטסט שלו ומתוך התיקייה שמכילה את קבצי הקוד של הסעיפים ואת תיקיית הטסטים:

```
./run_test.sh ex1.asm ex1sample_test
```

<u>**הערה**</u>: ייתכן ולפני הרצת קבצי sh על המכונה, תצטרכו להריץ את הפקודה –

```
chmod +x <your .sh file>
```

כתיבת טסטים בעצמכם

לכל תרגיל תוכלו לכתוב טסט, שהמבנה שלו דומה למבנה של `ex1sample_test`, עם שינוי תוויות ובדיקות בהתאם.

תרגיל 1 - shift key (12 נק')

עליכם לממש את ex1 המוגדרת בקובץ ex1.asm.

בתרגיל זה תקבלו תווית בשם `character`, המכילה תו (קוד ASCII) בגודל byte אחד.

עבור קלט זה, עליכם לחשב את קוד ה-ASCII שמתקבל בלחיצה על לחצן shift במקלדת ביחד עם הלחצן של תו המקור. באופן מדויק יותר, ניתן לראות בנספחים, שבסוף המסמך, את הטבלה של התווים שמצופה מכם להפעיל עליהם shift, ואת התוצאה המצופה, כאשר עליכם לפעול לפי ההוראות שלהלן:

- העמודה השמאלית מכילה את התווים שעליהם תצטרכו לבצע שינוי ("תו מקורי").
- עבור קבלה של תו בצד ימין של הטבלה, עליכם לשים את התו עצמו.
- עבור כל תו שלא בטבלה (= לא במקלדת) עליכם לשים `0xff` ב-shifted.

את התוצאה עליכם לשים בתווית בשם shifted, שגודלה byte אחד.

לתרגיל זה יש טסט לדוגמא, כמו שצוין מוקדם יותר במסמך. אם אתם קוראים על זה פעם ראשונה, סימן שלא עברתם על כל המסמך, וסביר שתקבלו בין 0 ל20 על כל התרגיל אלא אם כן תקראו את כל המסמך.

<u>שימו לב: על תרגיל 1 לא יתקבלו ערעורים מסוג תיקון קטן בקוד – הוא אולי ארוך ומציק, אבל אין סיבה שלא תבדקו תו תו ותבדקו שעובד</u>

תרגיל 2 – מעגל בגרף מכוון (16 נק')

עליכם לממש את ex2 המוגדרת בקובץ ex2.asm.
בתרגיל זה תקבלו גרף **מכוון** שמיוצג ע"י רשימת צמתים ורשימת שכנים בצורה הבאה:

- vertices – תווית שבה מערך של מצביעים לצמתים (מצביע בקורס, אם לא נאמר אחרת, הוא בגודל 8 בתים). הערך 0 מסמן את סוף המערך. כלומר אם יש בגרף 5 צמתים, יהיו במערך 6 איברים, כאשר האחרון הוא 0. ניתן להניח כי כל המצביעים במערך מכילים כתובות ב-section data.
- כל צומת (תווית כלשהי ב-section data שאינה ידועה מראש) הוא מערך של מצביעים לצמתים, שקיימת אליהם קשת מכוונת מהצומת. גם כאן 0 יסמן את סוף המערך.

בנוסף, תקבלו את התווית circle בגודל בית אחד. עליכם לשים בתווית circle את הערך 1 אם יש בגרף מעגל מכוון, ואת הערך 1- אחרת.

הערה: אפשר להניח שיש לכל היותר 8 צמתים.

תרגיל 3 – פיצול מערך למערכים ממוינים (18 נק')

עליכם לממש את ex3 המוגדרת בקובץ ex3.asm.

בתרגיל זה תקבלו את ארבע התוויות הבאות:

- `size` – מכילה ערך מטיפוס `unsigned int` (4 בתים).
- `source_array` – מערך באורך `size` של ערכים מטיפוס `int`.
- `bool` – תווית יעד (1 byte).
- `up_array, down_array` – שתי תוויות יעד.

מצופה מהקוד שלכם לפצל את `source_array` למערך אחד יורד ממש ומערך אחד עולה ממש, בהתאמה, בלי לשנות את הסדר המקורי, אם ניתן.

אם הדבר אפשרי, בנוסף שימו בתווית bool את הערך 1. אם הדבר לא אפשרי, שימו בה את הערך 0.

דוגמאות:

1. עבור מערך מקור (1, -2, 3, -4) קיימות כמה תשובות אפשריות:
   - ♪ (1, 3) ו- (-2, -4)
   - ♪ (1, -2, -4) ו- (3)
   - ♪ (1, -4) ו- (-2, 3)
2. עבור המערך (1, 5, -3, -1) לא קיים פתרון.

הנחות והצעות:

1) מומלץ לכתוב פסאודו אלגוריתם (ניתן לפתור ב-$O(n)$), ולוודא שהוא עובד על הדוגמאות לפני שפונים למימוש באסמבלי. האלגוריתם הסופי לא אמור להיות מסובך והקוד לא אמור להיות ארוך מדי. **בפרט** - אל תנסו להתחכם ולהבין מראש שלמערך מסוים אין פתרון, למשל.

2) אין צורך לדאוג מכך שיהיו כמה פתרונות ורק חלק מהם יעברו את הטסט. או שכל הפתרונות יעברו, או שיהיה רק פתרון אחד.

3) קראו שוב את ההוראות וההנחות הכלליות בתחילת המסמך.

תרגיל 4 – סדרה ברשימה (24 נק')

עליכם לממש את ex4 המוגדרת בקובץ ex4.asm.
בתרגיל זה תעבדו עם רשימה מקושרת. כל איבר ברשימה זה יורכב משני שדות, כמתואר בהגדרת ה-struct הבאה:

```
struct Node {
    long data;
    struct Node *next;
}
```

רמז 1: struct Node זה מצביע.
רמז 2: `long` זה `unsinged quad`.
רמז 3: ניתן לראות דוגמה לרשימה בתרגול 3.

בתרגיל זה תקבלו את שתי התוויות הבאות:

- `head` – מצביע (8 בתים) לאיבר הראשון ברשימה.
- `result` – תווית יעד בגודל בית אחד.

הניחו שהרשימה מייצגת סדרת מספרים כך שהאיבר הראשון בסדרה הוא ה-data של האיבר הראשון, וכך הלאה.

בתרגיל זה עליכם לשים בתווית result את הערכים 0, 1, 2, או 3, לפי התנאים הבאים:

- 3 – אם הסדרה עולה ממש
- 2 – אם היא עולה (לא יורדת).
- 1 – אם היא כמעט עולה [1]
- 0 – אחרת.

רמז 4: אל תנסו לבדוק את כל המקרים במעבר אחד על הרשימה.

[1] סדרה היא כמעט עולה אם הסרת איבר אחד בדיוק מהרשימה יהפוך אותה לעולה. שימו לב שאין משמעות ל "איבר אחד או פחות" כי פחות אומר שהיא בעצם לא יורדת.

תרגיל 5 – atoi (30 נק')

עליכם לממש את ex5 המוגדרת בקובץ ex5.asm.

בתרגיל זה תקבלו את שלוש התוויות הבאות:

- `command` – מחרוזת null terminated.
- `legal` – תווית יעד בגודל byte אחד.
- `integer` – תווית יעד בגודל 4 bytes.

המחרוזת `command` היא פקודה באסמבלי. עליכם לבדוק האם יש אופרנד מסוג קבוע חוקי בפקודה (אין צורך לבדוק שהפקודה עצמה חוקית), כאשר קבוע חוקי בשאלה הוא קבוע מספרי בלבד (בפרט, לא label).

ולשים 1 בתווית `legal` אם כן ו 0 אחרת. כמו כן אם שמתם 1, יש לשים ב-integer את המספר.

הנחיות, הנחות, הסברים, ורמזים:

- קבוע חוקי מתחיל ב$.
- עליכם לזהות את הבסיס בו הקבוע מיוצג ע"פ התווים שאחרי ה$.
- לא יהיה מינוס במספר.
- ניתן להניח שאחרי התווים (במידה וישנם) שמייצגים את בסיס הספירה, יופיעו 3 תווים לכל היותר. השתמשו בנתון זה והמנעו ממימוש של חזקה בלולאה! בפרט, חשבו מראש את החזקות והכפילו כמה שפחות.
- ניתן להניח שאם קיים אופרנד קבוע הוא תמיד הראשון ותמיד בפקודה בת 2 אופרנדים או יותר – כלומר, בסוף האופרנד יגיע פסיק (ללא רווחים).
- אין הנחות נוספות. כל מקרה קצה צריך להיבדק וכנראה יוביל או ל 0 ב legal או ל 0 ב integer.

מילים אחרונות

איך בונים ומריצים לבד?

כאמור בתחילת התרגיל, נתון לכם קובץ טסט לתרגיל 1 וקובץ run_test.sh. אתם יכולים לשנות את הקובץ של תרגיל 1 ולהשתמש במבנה שלו כדי לבדוק תרגילים אחרים באותה הצורה, כפי שהוסבר קודם לכן בהקדמה לתרגיל.
כדי להריץ, או לנפות שגיאות:

```
as ex1.asm ex1sample_test -o my_test.o  #run assembler (merge the 2 files into one asm before)

ld my_test.o my_test.o -o ex1           #run linker

./ex1                                   #run the code

gdb ex1                                 #enter debugger with the code
```

את הקוד מומלץ לדבג עם gdb. לא בטוחים עדיין איך? על השימוש ב-gdb תוכלו לקרוא עוד במדריך באתר הקורס.

בטסט אתם תפגשו את השורות הבאות

```
movq $60, %rax
movq $X, %rdi          # X is 0 or 1 in the real code
syscall
```

שורות אלו יבצעו exit(X) כאשר X הוא קוד החזרה מהתוכנית – 0 תקין ו-1 מצביע על שגיאה. בקוד שאתם מגישים, אסור לפקודה syscall להופיע. קוד שיכיל פקודה זו, יקבל 0.

SASM

שימו לב: למכונה הוירטואלית של הקורס מצורפת תוכנת sasm, אשר תומכת בכתיבה ודיבוג של קוד אסמבלי וכן יכולה להוות כלי בדיקה בנוסף לgdb. (פגשתם אותה בתרגיל בית 0). כתבו בטרמינל:

```
sasm <path_to_file>
```

כדי להשתמש ב-SASM לבנייה והרצת קבצי ה-asm, עליכם להחליף את שם התווית _start בשם main (זאת מכיוון ש-sasm מזהה את תחילת הריצה על-ידי התווית main. **אל תשכחו להחזיר את _start לפני ההגשה!).**

ניתן גם לדבג באמצעות מנגנון הדיבוג של SASM במקום עם gdb, אך השימוש בו על אחריותכם (**כלומר לא נתמך על ידנו ולא נתחשב בבעיות בעקבותו בערעורים**. שימו לב לשוני בין אופן ההרצה ב-SASM לאופן ההרצה שאנו משתמשים בו בבדיקה שלנו).

נספחים

טבלת המרות מצופות לתרגיל 1: (סה"כ 47 המרות)

תו מקורי	תו shifted
1	!
2	@
3	#
4	$
5	%
6	^
7	&
8	*
9	(
0	)
`	~
-	_
=	+
[	{
]	}
;	:
'	"
\	|
,	<
.	>
/	?
a-z	A-Z