תרגיל בית 1 – הגשה ב-03/02/2024

הערות להגשה:

- את העבודה יש להגיש בזוגות בלבד.
- יש להגיש את התרגיל אך ורק מהמודל של **מאחד מבני הזוג** (אל תגישו שניכם בנפרד!).
- יש להגיש סה"כ 6 קבצים בדיוק (כלומר, 6 קבצים בלבד ולא zip שלהם):
  - קובץ asm עבור כל סברוטינה בנפרד:
    בהמשך יוגדר עבור כל סברוטינה שתדרשו לכתוב את הכתובת בזיכרון בה יתחיל מימוש הקוד.
  - קובץ submitters.txt עם הת"זים ואחוז ההשתתפות בהתאם לפירוט בהמשך.
- עליכם לציין בקובצי ה-asm וה-txt את תעודות הזהות ואת אחוז העבודה של כל שותף, באופן הבא בלבד:

; 123456, 30%
; 456789, 70%

שימו לב לרווחים, פסיקים ונקודה פסיק שבתבנית לעיל. אל תחרגו ממנה!
יתרה מזאת, בקבצי ה-asm, אין לבצע מרווח של שורה או יותר בין שורות הת"ז לבין שורת ה-orig.
יש דוגמה המצורפת בעמוד האחרון של קובץ ההוראות שתראה את המבנה.
שימו לב, שותף אשר לא יהיה פעיל בתרגיל הבית עלול להיקנס בנקודות.
תהיה השגחה על כך ואף תיתכן בדיקה מדגמית של הבקיאות שלכם בפתרון.

- ניתן להגיש באיחור של עד ארבעה ימים ללא עדכון צוות הקורס. כל יום איחור יגרור 5 נקודות הורדה בציון.
  לא תתקבלנה הגשות לאחר ארבעה ימים ממועד ההגשה.
- אנא הקפידו על פתרון נקי, אלגנטי ומתועד לכל אורכו, אשר יסביר את הלוגיקה שלכם עם שמות לייבלים משמעותיים. שימו לב – זהו קוד אסמבלי, לכן תיעוד טוב אינו רשות, אלא **חובה!**
- על שמות הלייבלים של הסברוטינות/פונקציות שלכם להיות **בדיוק**, אחד לאחד, כפי שנכתב בתרגיל זה, יש לשים לב ל-case sensitivity של הלייבלים (באופן כללי).
- שימו לב, הטסטים נבדקים אחד אחרי השני ללא איפוס הסימולטור.
- **בדיקת תרגילי הבית הינה חצי אוטומטית. אי עמידה בדרישות הנ"ל עשויה לגרור הורדה בציון!**
  - אי עמידה בדרישות הפורמט אשר כולל:
    - שורות תעודת הזהות
    - שמות הקבצים

    תגרור הורדה של נקודות מסך הציון של תרגיל הבית.
  - הדפסות של התרגיל צריכות להיות בהתאם לתבנית המוצגת בתרגיל.
  - **כל שורת פלט אשר אינה זהה** אחד לאחד לפלט המצופה **תגרור הורדה של נקודה מהטסט.**
- שאלות בנוגע לתרגיל הבית יש להפנות לפורום.
- בעניינים אישיים ניתן לפנות למתרגל במייל:

AlonPapiniCSTA@gmail.com

בתרגיל זה נממש מספר סברוטינות בסיסיות ב-LC3.

1) סברוטינה בשם **Mul** המבצעת כפל בין שני רגיסטרים:
הסברוטינה Mul מכפילה את שני הערכים שברגיסטרים R0 וב-R1, ושמה את התוצאה ב-R2.
על הפתרון לכלול טיפול במספרים שליליים ואפסים.
על הסברוטינה לתמוך בתוצאות הבאות לפחות:

Mul(0, 1666) = 0
Mul(915, -16) = -14640
Mul(-11, 555) = -6105
Mul(-12, -12) = 144
Mul(12, 123) = 1476

ממשו את הסברוטינה מכתובת X4000.

2) סברוטינה בשם **Div** המבצעת חלוקה בין שני רגיסטרים:
הסברוטינה Div מחלקת את הערכים שברגיסטרים R0 וב-R1, ושמה את תוצאת החלוקה ברגיסטר R2 ואת השארית ברגיסטר R3.
החלוקה $\frac{R0}{R1}$ תבוצע ע"י החסרה איטרטיבית של המחולק מהמחלק. לדוגמא ,חלוקה של 22 ב-10 תבוצע ע"י החסרה של 10 מ-22 פעמיים, כך שהשארית שנותרה הינה 2 ותוצאת החלוקה הינה 2.
תוצאת החלוקה היא כמות הפעמים שבוצעה פעולת החיסור, והשארית הוא המספר שנותר בין 0 למחלק.
על הפתרון לכלול טיפול במקרים של חלוקה של מספרים שליליים ואפסים.
עבור קלט לא חוקי, כמו חלוקה באפס, יש להחזיר 1- הן במנה והן בשארית.
על הסברוטינה לתמוך בתוצאות הבאות לפחות (המספר משמאל הוא המנה, מימין זה השארית):

Div(0, 25) = 0 0
Div(25, 0) = -1 -1
Div(-110, 3) = -36 2
Div(110, -3) = -36 2
Div(28, 9) = 3 1

ממשו את הסברוטינה מכתובת X4064.

3) סברוטינה בשם **Exp** המבצעת העלאה בחזקה בין שני רגיסטרים:
הסברוטינה Exp עבור הערכים שברגיסטרים R0 וב-R1 מבצעת $R0^{R1}$, ושמה את התוצאה ב-R2.
הפעולה $R0^{R1}$ תבוצע ע"י שימוש בסברוטינה Mul.
על הפתרון לכלול טיפול במקרים מיוחדים, כגון מספרים שליליים או אפסים.
עבור קלט לא חוקי יש להחזיר 1-. מהם קלטים לא חוקיים? להלן הפירוט:

- $R0 = 0 \;\&\&\; R1 \leq 0$ (תנאי של וגם)
- $R0 < 0 \;||\; R1 < 0$ (תנאי של או)

על הסברוטינה לתמוך בתוצאות הבאות לפחות:

Exp(1,12) = 1
Exp(1,-12) = -1
Exp(0,122) = 0
Exp(0,0) = -1
Exp(1,0) = 1
Exp(4,2) = 16

ממשו את הסברוטינה מכתובת X40C8.

<u>הערה חשובה</u>: Exp משתמשת ב-Mul, אבל אתם **לא** מממשים את Mul באותו הקובץ, לכן אם סתם תכתבו JSR Mul תקבלו שגיאה! מה שאתם כן צריכים זה להשתמש בפקודה JSRR, ולהטעין לרגיסטר מסוים את הכתובת של Mul...

4) סברוטינה בשם **CheckPrime** שבודקת האם מספר אי-שלילי (מותר להניח כך!) הוא ראשוני.
המספרים יהיה ב-R0, והסברוטינה תחזיר ב-R2 את המספר 1 אם המספר אכן ראשוני, ו-0 אם לא.
שימו לב ש-0 ו-1 הם מספרים אי-שליליים, ולשם נוחות תוכלו לתת להם מענה מיוחד (כאשר אתם יודעים שהם אינה, ע"פ הגדרה, מספרים ראשוניים), אבל החל מ-2 על הפתרון שלכם להיות כבר כללי!

<u>רמז</u>: סביר לחשוב שתרצו לבדוק אם מספר כלשהו בין 1 למספר הנתון עצמו מחלק את המספר הנתון עם שארית 0. נשמע שצריך להיעזר פה ב-DIV בלולאה כלשהי...

על הסברוטינה לתמוך בתוצאות הבאות לפחות:

CheckPrime(0) = 0
CheckPrime(1) = 0
CheckPrime(2) = 1
.
.
.
CheckPrime(6) = 0
CheckPrime(7) = 1
.
.
.
CheckPrime(23) = 1
.
.
.

ממשו את הסברוטינה מכתובת X412C.

5) סברוטינה בשם **TriangleInequality** הבודקת האם אי-שוויון המשולש מתקיים בין שלושה מספרים:
הסברוטינה תשים ב-R3 את הערך 1 אם הערכים המספריים של R2, R1, R0 מקיימים את אי-שוויון המשולש ו-0 אחרת, כאשר עבור קלט לא חוקי יש להחזיר 1-.
קלט לא חוקי זה ערכים שליליים, מספיק לפחות כי ערך אחד מתוך השלשה שיהיה שלילי.
אי-שוויון המשולש מתקיים אם סכום כל זוג מספרים גדול או שווה למספר השלישי, עבור כל הזוגות האפשריים.
כלומר, $R0 \leq R1 + R2$ וגם $R1 \leq R0 + R2$ וגם $R2 \leq R0 + R1$.
לדוגמה, שלשת המספרים 1,2,4 אינה מקיימת את אי-שוויון המשולש, כיוון ש- $4 \nleq 1 + 2$.
לעומת זאת השלשה 2,3,4 אכן מקיימת את אי-שוויון המשולש.
יש לתמוך הן במספרים חיוביים, הן במספרים שליליים והן באפסים עבור $R0, R1, R2$.
על הסברוטינה לתמוך בתוצאות הבאות לפחות:

TriangleInequality(6, 9, 10) = 1
TriangleInequality(1,5,8) = 0
TriangleInequality(1,2,-3) = -1

ממשו את הסברוטינה מכתובת X4190.

<u>הסבר חשוב על מבנה התרגיל</u>: תרגיל בית זה אינו מוגש בקובץ אחד, אלא בכמה קבצים נפרדים. לדוגמה, על מנת לבדוק את תקינות פעולת Exp שתממשו עליכם לפתוח את הסימולטור ולהטעין לו **גם** את הקובץ של Mul **וגם** את הקובץ של Exp. כלומר, אתם טוענים יותר מקובץ אחד לסימולטור בבת אחת (ולמעשה כך גם תרגיל הבית שלכם ייבדק – כל הקוד שלכם ייטען לסימולטור ובנוסף גם קוד הטסטים עצמם, שבעצמו יתחיל בכתובת X3000). שימו לב לדקות ולסיבוכיות הנוספת שבכתיבה, בבדיקה ובהרצת תרגיל הבית כתוצאה מכך.

הצעה למבנה מעטפת לבדיקת הקוד:

ניתן לוודא כי הסברוטינות אשר כתבתם עובדת כנדרש באמצעות כתיבת מעטפות בדיקה.

לנוחיותכם, דוגמה למעטפת עבור הסברוטינה Mul (שניתנת לכם בקובץ main_test_example.asm):

```
;Test program for integer multiplication
.ORIG x3000
LD R0, Test_Mul1                        ; R0 = Test_Mul1
LD R1, Test_Mul2                        ; R1 = Test_Mul2
LD R3, MulFuncPtr                       ; R3 = The address of Mul
JSRR R3                                 ; R2 = Mul(R0, R1)
LD R1, Test_Res                         ; R1 = Test_Res
; At this point R1 holds (-1) * correct answer
; While R2 holds the result that the function returned
ADD R2, R2, R1                          ; R2 = R2 + R1, and remember that R1 is (-1) * The answer!
BRz RES_GOOD                            ; If R2 = 0 then jump to Res_Good else continue to Res_Bad
RES_BAD:
        LEA R0, TEST_ERR_STR
        PUTS
        BR DONE
RES_GOOD:
        LEA R0, TEST_CORRECT_STR
        PUTS
DONE:
        HALT                            ; Program is done and will now exit
TEST_ERR_STR .STRINGZ "Result is wrong"
TEST_CORRECT_STR .STRINGZ "Result is correct "
Test_Mul1 .FILL #6
Test_Mul2 .FILL #70
Test_Res .FILL #-420                            ; (-1)*6*70
MulFuncPtr     .FILL X4000              ; Pointer to Mul subroutine : R2 <-- R0 * R1
.END
```

תוכלו לכתוב מעטפות בדיקה דומות עבור הסברוטינות האחרות גם כן.

כל שעליכם להגיש הוא קובץ נפרד עבור כל סברוטינה שכתבתם, מבנה כל קובץ יהיה להלן:

```
; ID of student 1, __%
; ID of student 2, __%
.ORIG X__
```

Name of the subroutine (example "Mul:")

Write here your code!

RET

Write here your constant/labels!

.END