מבוא למערכות מחשוב – עבודה מספר 3

כללי – לעבודה זאת 2 חלקים עיקריים. בחלק הראשון תכתבו כמה תוכנות פשוטות באסמבלי. בחלק השני תממשו אסמבלר. העבודה קשה ומאתגרת, ורצוי להתחיל בה בהקדם האפשרי.

תאריך ההגשה – יום רביעי, ה-30 בדצמבר, בשעה 23:55. בהגשה יש לכווץ את את התיקיה כפי שניתנה לכם, ללא הסרת קבצים כלשהם, על ידי תוכנת כיווץ עם סיומת zip (לא rar,tar,7z ודומיהם), ולהגישה לשרת ה-ftp של המחלקה.

1. עליכם לתרגל תחילה מעט כתיבה באסמבלי.
   1. כתבו קוד אסמבלי המבצע הכפלה בין המספרים שבתא הראשון (R0) והשני (R1) בזכרון, וכותב את התשובה לתא השלישי (R2). עליכם ליצור תוכנית בשם Mult.asm, לתרגם אותה באמצעות האסמבלר המצורף, ולהריצה על המכונה.
   2. כתבו קוד אסמבלי המבצע פעולת חזקה R2 = R0R1. עליכם ליצור תוכנית בשם Power.asm.
   3. כתבו תוכנית הממלאת, על ידי שימוש בלולאה, את התאים מ-100 עד 119 בזכרון ב-20 האיברים הראשונים של סדרת פיבונאצ'י – (1,1,2,3,5,8,13,…). צרו תוכנית בשם Fibonacci.asm.
2. כעת אתם מוכנים לממש אסמבלר משלכם. הפרויקט Assembler מכיל שלד של מימוש תהליך התרגום, כולל פונקציות רבות העוסקות בקריאה מקבצים ובעבודה מול מחרוזות. ניתן לבדוק את הפלט שלכם מול הפלט של האסמבלר שסופק בחבילת הכלים המצורפת. כזכור, לתהליך התרגום יש שלושה שלבים – תרגום מקרו, מעבר ראשון ליצירת טבלת הסמלים, ומעבר שני ליצירת שפת המכונה. אנו נממש את השלבים בסדר הפוך:
   1. ממשו את השיטה SecondPass המתרגמת כל שורה בקוד למחרוזת המכילה בדיוק 16תווים מתוך {0,1} ומייצגת מספר בקוד בינארי, ללא התיחסות לסמלים. שלד השיטה כבר נתון לכם. את השיטות המתורגמות הכניסו לרשימה lAfterPass על ידי lAfterPass.Add(s). לאחר שסיימתם תוכלו להפעיל את האסמבלר על כל קבצי הקוד שאינם מכילים סמלים: Add, MaxL.
   2. הוסיפו מבנה נתונים (Dictionary) המייצג את טבלת הסמלים. ממשו את השיטה FirstPass היוצרת וממלאת את טבלת הסמלים – משתנים ותוויות. שנו את השיטה SecondPass כך שתתמוך בתרגום סמלים למספרים. לאחר שסיימתם, תוכלו להריץ את כל תוכנות האסמבלי שאינן מכילות מקרו.
   3. ממשו את השיטה ExpandMacro המקבלת שורת קוד אסמבלי בודדת העשויה להכיל מקרו ומחליפה אותה במספר שורות הממשות את המקרו. במידה ובשורה אין מקרו, יש לרשום לרשימת ההחזרה את השורה המקורית. עליכם לתמוך במקרו הבאים:
      1. direct addressing, לדוגמא – M[x]=D או A=M[y]. למעשה עליכם להחליף כל מופע של M[x] בשורות הממשות אותו. אין צורך לתמוך בשימוש כפול בdirect addressing כגון M[x]=M[y].
      2. קיצורים לקפיצות, לדוגמא – D;JGT:LOOP – מאפשר לרשום את יעד הקפיצה באותה שורה של פקודת הקפיצה.
      3. לאחר מימוש המקרו האלו תוכלו לתרגם את SquareMacro.asm.

בדקו את קוד שפת המכונה שיצרתם על גבי המכונה שלכם, או על גבי הסימולטור הנתון ב-tools.zip. וודאו שהתוכניות רצות כיאות ומוציאות את הפלט הנדרש.

לעבודה מצורפות תוכנות עזר בתיקיה tools.zip, הממשות, בין השאר, אסמבלר ומדמה (סימולטור) של התנהגות ה-CPU, שיאפשרו לכם לתרגם ולהריץ כל קוד אסמבלי שתרצו על מנת לבדוק האם המכונה והאסמבלר אותם בניתם מתפקדים כראוי.

אתיקה – את התרגיל יש לעשות לבד (לא בזוגות). מותר להתיעץ עם סטודנטים אחרים, אך לא להעתיק קטעי קוד מסטודנטים או מהאינטרנט. כל שליחה או קבלה של קטע קוד ממקור כלשהו – סטודנט אחר או מהאינטרנט, במייל, בדיסק נייד, או דרך אתרי שיתוף קבצים אסורה בהחלט! אל תעתיקו!

שאלות ובעיות יש לפרסם בפורום. במידה ולא תקבלו תשובה, יש לפנות לגיא במייל [shanigu@bgu.ac.il](mailto:shanigu@bgu.ac.il)

בהצלחה!

.