סקירה כללית על פרויקט גמר
קורס "מעבדה בתכנות מערכות", 2023ב

מגישים: אלרואי הלל 207217126
טמיר כהן 209199041

מבוא

הפרויקט מדמה פעולת אסמבלר על שפת אסמבלי שהוגדרה ע"י צוות הקורס. דף הסבר זה והתיעוד שבקוד עצמו יעסוק בהסבר על פעולת האסמבלר, אך אנו מניחים כי שפת האמסבלי עצמה מוכרת לקורא הקוד על כל פרטיה. מטרת דף הסבר זה היא להסביר את המהלך הכולל של האסמבלר, כדי לספק תמונה מלאה, שמטבע הדברים עלולה להיבלע בתוך הקוד הרב (כמאמר הפתגם "מרוב עצים לא רואים את היער..."). פרטי הקוד והסבר על כל פונקציה- יופיעו בתיעוד שבקוד.

פעולת האסמבלר מנוהלת ע"י שלושה חלקים עיקריים, אנו נפרט בקצרה את מטרתו של כל אחד:

א) **פרה אסמבלר:** מטרתו העיקרית היא "לנקות" את הקוד עבור האסמבלר עצמו: לפרוס מאקרויים ולמחוק שורות הערה ושורות ריקות.
ב) **מעבר ראשון:** ממלא שתי מטרות עיקריות: הראשונה, בדיקת תקינות הקוד. השנייה, בניית טבלת הסמלים. רק אם הקוד התגלה כתקין- האסמבלר ימשיך בפעולתו.
ג) **מעבר שני:** מטרתו העיקרית היא להמיר את קוד האמסבלי לשפת מכונה. הוא משרת גם מטרה משנית, והיא יצירת הטבלה של התוויות החיצוניות.

לבסוף מתבצעת גם **כתיבת הטבלאות החיצוניות והפנימיות:** ייצוא 2 קבצים: הראשון, טבלה של תוויות שהוגדרה כפנימיות. השני, טבלה של תוויות שהוגדרו כחיצוניות.

החלקים הללו כמובן משתמשים בפונקציות עזר, שאמנם אינן מריצות את האסמבלר- אבל מבצעות פעולות משמעותיות בו:

- **Label**- קובץ שמכיל את הפונקציות המאפשרות לטפל, להוסיף ולעדכן את טבלאות הסמלים והנתונים.
- **CutPaste**- "גזור הדבק" קובץ שמכיל פונקציות שמאפשרות לקחת את הנתון הרצוי מתוך שורת הקוד "האנונימית", כדי לזהות אותו, לטפל בו וכן הלאה.
- **Identify**- קובץ המכיל פונקציות שעניינן לבצע זיהוי על מחרוזת ספציפית (שלרוב התקבלה מהפונקציות של CutPaste)- לברר במה מדובר? (בפעולה, בהנחיה, באופרנד...) והאם הוא תקין?
- **BinBase64**- קובץ המכיל פונקציות העוסקות בהמרת קוד לשפת מכונה- תחילה לבינארי ואז לבסיס 64. קובץ זה משמש את המעבר השני בלבד.

המשתנים הגלובליים

השתדלנו למעט במשתנים גלובליים ככל שניתן, עם זאת מאחר והשימוש בהן הותר במידה סבירה בפרויקט- בחרנו בכמה משתנים גלובליים שהיו נראים בעינינו כהכרחיים ותורמים באופן משמעותי להרצת הקוד:

error- משתנה דגל, שכל פונקציה שבודקת תקינות יכולה להרים ולהודיע: "נמצאה שגיאה". מרגע שנמצאה שגיאה פעולת האסמבלר לא תימשך. חיוני שתהיה גישה מכל הקבצים למשתנה זה.

ic, dc- אלו המונים של שורות קוד המכונה, הם משמשים את המעבר הראשון בבניית טבלת הסמלים, וכן מודפסים בקובץ קוד המכונה.

טבלאות הסמלים- טבלה המשמשת לאורך כל פעולת האסמבלר, לכל הפונקציות נדרשת גישה אליה.

פרה אסמבלר

פרה האסמבלר מטרתו היא להביא את הקובץ למצב שכל שורה היא רלוונטית לתרגום לקוד הסופי. לשם כך הפרה אסמבלר מבצע שתי פעולות כלליות.

האחת: ניקוי כל השורות הריקות המופיעות בקובץ, וכן שורות ההערה המסומנות ';' בתחילתן. בחרנו לבצע את פעולות אלה בפרה האסמבלר למרות שלא נכתב במפורש, משום שמחיקת שורות אלו צריכה להיעשות עוד בשלב שאיננו מתייחסים לכתיב הטקסט לגופו, ולכן הסקנו שפעולה בסיסית זו צריכה להיעשות בשלב הראשון.
השנייה: פרישת קוד המקרו המתאים כולל ווידוא שהמקרואים מוגדרים נכון והדפסת הודעות שגיאה בהתאם.

התוכנית תכין קובץ חדש עם סיומת am. ותדפיס אליו את השורות החדשות.

התוכנית תרוץ על שורות הקוד ותמצא היכן מוצהר מקרו, תוודא את תקינות הכתיב ותיצור מקרו חדש. את המקרואים התוכנית תשמור בתור רשימה מקושרת של משתנים מסוג מקרו המכילים שמות ותוכן.

לאחר יצירת הרשימה המקושרת התוכנית תרוץ על קובץ המקור תדלג על שורות ההגדרה של המקרואים, ההערות והשורות הריקות, ותזהה כאשר קיים שם של מקרו לפרישה ותבצע את הפרישה לקובץ החדש.

טבלת הסמלים

את טבלת הסמלים בחרנו לממש בצורה של רשימה מקושרת, כדי לאפשר דינמיות בהכנסת התוויות אל הטבלה. האסמבלר מנהל 3 טבלאות: טבלת סמלים, טבלת נתונים, טבלת תוויות חיצונית. טבלת הנתונים היא למעשה שדה עבור כל תווית בטבלת הסמלים. טבלת התוויות החיצוניות תיבנה במעבר השני.

טבלת הסמלים הראשית, מכילה את כל הנתונים על התווית- שמה, הכתובת שלה, האם היא תווית מקור או נתונים, והאם היא הוגדרה כחיצונית או פנימית. במידה וזו תווית נתונים- היא תצביע על טבלת נתונים המיוחדת לה. יש פונקציות שנועדו לאפשר להוסיף אליה תווית, לחפש תווית ולעדכן נתוני תווית. לעומתה, טבלת הנתונים דורשת פונקציה אחת של הוספה. בכל פעם שנרצה לגשת לנתונים- ניגש אליהם דרך התווית שהם שייכים אליה.

רק לאחר סיום המעבר הראשון אנו יכולים לדעת את כל התוויות החיצונית, ולכן רק במעבר השני נבנה את הטבלה החיצונית: בכל פעם שהיא תיפגש בקריאה לתווית שהיא חיצונית- היא תכתוב את הכתובת ממנה התבצעה הקריאה. בסוף פעולת האסמבלר, תודפסנה הטבלאות החיצונית והפנימיות בעזרת פונקציות ייעודיות.

טבלת הנתונים משתמשת בהקצאה דינמית- ויש 2 נקודות יציאה בהן הזיכרון יכול להשתחרר: בקוד תקין, הזיכרון ישתחרר בסוף פעולת האסמבלר- כאשר הטבלאות הפנימיות והחיצוניות יודפסו. אם הקוד התגלה כלא תקין, אנו נשחרר את הזיכרון לאחר המעבר הראשון, לפני סיום התוכנית.

הקובץ CutPaste

על מנת לבצע עיבוד לשורה אנו נדרשים לפרק את השורה ולדלות ממנה את המידע הדרוש לנו לשם הבדיקות והתרגום, לשם כך הקובץ מכיל פונקציות שמבצעות פעולות מסוג זה, כגון הסרת התווית בתחילת השורה על מנת שיהיה ניתן לגשת בעזרת פונקציה אחרת למילה הראשונה בתווית ובדוק האם מדובר בשורת פעולה או נתונים או לחילופין לבדוק האם הכתיב תקין. כמו כן הקובץ מכיל פונקציות המחזירות את האופרנד הראשון והשני לשם מטרות דומות. רק פסיק הוא התו שמפריד בין אופרנדים, ולכן שתי מילים שאין ביניהן פסיק- יחשבו לאופרנד אחד. במצב זה, תשלח המחרוזת "אופרנד לא מזוהה"- והאופרנדים לא יחלו את תהליך הזיהוי.

הקובץ Identify

הפונקציות האלו לרוב יקבלו כפרמטרים את הערכים המוחזרים מהפונקציות שבקובץ CutPaste. הקובץ מחולק לשני נושאים מרכזיים: הראשון, פעולות ואופרנדים- נבדקת התקינות של כל אחד בפני עצמו וההתאמה ביניהם. השני, הנחיות ונתונים.

לאורך כל הקובץ שזורות שתי פעולות שהן למעשה כמעט אחת: זיהוי ותקינות. ההבדל בין זיהוי לתקינות הוא דק, ולכן קבענו "כללי אצבע" לפיהן נקבע שהתבצע זיהוי, ולאחר מכן נבדוק את תקינותו. כללים אלו מפורטים היטב בתיעוד הקובץ. לדוגמה: אופרנד שמתחיל בתו @, יחשב כרגיסטר, לאחר מכן נבדוק האם הוא רגיסטר תקין- ואם לא נודיע שהרגיסטר לא תקין. כמו כן, הוראה שתתחיל בנקודה- תחשב להנחיה, וכן הלאה כפי שניתן לקרוא בקובץ.

המעבר הראשון

המטרה הראשונה- בדיקת תקינות הקוד: המעבר הראשון לוקח שורה אחר שורה מהקובץ שיצר הפרה אסמבלר, נוטל ממנה חלק אחר חלק בעזרת הפונקציות מהקובץ CutPaste, ובהתאם להקשר- קורא לפונקציות מהקובץ Identify לבצע זיהוי ותקינות.

המטרה השנייה- בניית טבלת הסמלים: במידה והקוד זיהה שורה תקינה, המכילה הוראה הקשורה לתווית- הוא יוסיף או יעדכן את התווית בטבלת הסמלים (הדברים בקוד מעט יותר מורכבים ממה שמתואר כאן, אך זו התמונה הכללית). את הכתובות הוא מונה בעזרת המונים ic, dc- אשר מתעדכנים באופן חי ודינמי תוך כדי התוכנית, ומספקים את תמונת קוד המכונה העתידית עבור כל שורה ושורה- וממילא עבור הכתובות של התוויות. הסבר מפורט יותר על מונים מופיע בתיעוד הקובץ.

בסוף המעבר הראשון, הכתובות בטבלת הסמלים הן גולמיות, ומנוהלת במקביל ע"י 2 מונים שונים. לכן, מתבצע מעבר על טבלת הסמלים שמעדכן את התוויות לכתובות האמיתיות שלהן: לפי 2 קווים מנחים: הראשון, נקודת ההתחלה של הזיכרון (במקרה זה היא 100). השני, שבקוד המכונה ההוראות והנתונים מחולקים לשני חלקים שונים, כאשר ההוראות קודמות לנתונים.

המעבר השני

אנו נקרא למעבר השני, רק אם המעבר הראשון עבר תקין- כלומר, הקוד זוהה כתקין ע"י האסמבלר. המעבר השני מבצע למעשה שני מעברים על הקוד: בפעימה הראשונה, הוא מלקט הוראות ומדפיס אותן בקובץ קוד המכונה. ובפעימה השנייה, הוא מלקט הנחיות נתונים ומדפיס את הנתונים בחלק השני של קובץ קוד המכונה.

הדפסת ההוראות- מחולקת לשני חלקים: החלק הראשון, אפיון שורת הקוד בהתאם לצרכים של קוד המכונה. האפיון מתעסק הן בשורת הפעולה עצמה, והן בשורות האופרנדים שיבואו אחריה. האפיון בסופו של דבר מתכנס לתוך משתנים. תהליך האפיון והמשתנים המכילים את המסקנות מפורט בתיעוד שבקוד. בחלק השני, אנו קוראים לפונקציות הממירות את השורה לשפת מכונה, שולחים אליהם את האפיון שנעשה- והן מצידן ממירות ומדפיסות את השורה בשפת מכונה. מאחר ולא נדרשת הדפסה בשפה הבינארית, אלא בבסיס 64- ההמרה הבינארית ולאחר מכן המרה לבסיס 64- נעשות בהוראה מקוננת אחת.

הדפסת הנתונים- מתעניין בהנחיות נתונים בלבד. ברגע שזוהתה הנחיה כזו, אנו מזהים את התווית בה שמורים הנתונים, ניגשים לטבלת הנתונים שבתווית זו, ושולחים ממנה את הנתונים אל הפונקציות הרלוונטיות שתבצענה המרה לשפת מכונה.

הקובץ BinBase64

הקובץ מכיל פונקציות שמטרתן לשמש את המעבר השני בתרגום הקוד לBase64. פונקציה מרכזית אחת שמלווה את כל הקובץ מתרגמת מספר מבסיס דצימלי לבינרי. הפונקציה מקבלת מספר ומחרוזת (שבסופו של דבר תוחזר למשתמש), וכן משתנה המייצג את מספר הביטים בהם יוחזר המספר שתורגם מדצימלי לבינרי. בהמשך הקובץ ישנן פונקציות המתרגמות שורת פעולה ואת השורות הבאות אחריה, כגון שורה המייצגת שני רגיסטרים או אחד וכן פונקציה המקבלת מספר לתרגום ובנוסף את סוג המיעון ומחזירה מחרוזת בהתאם.

בסוף הקובץ ישנה פונקציה הממירה מספר בינרי בן שש ספרות למספר דצימלי, דבר הדרוש לנו לשם התרגום של שורות הקוד הבינרי באורך 12 ספרות לקוד המיוצג בBase64. ובסוף הקובץ יש את הפונקציה המייצרת את הטבלה המתאימה לכל מספר את התו המתאים לו לפי Base64, פונקציה זו מקבלת מחרוזת בינרית באורך 12 מפצלת אותה לשניים ממירה אותה לBase64 לפי הטבלה ומחזירה את התוצאה.