קובץ דוקומנטציה לפרויקט –

מגישים

מור יגל
ארבל יניב

תיאור קצר של התוכניות וכיצד להריץ אותן

ישנן שתי תוכניות בסיסיות – asm וsim האם האסמבלר והסימולטור. בנוסף, ישנן 3 תוכניות נלוות – addmat.asm, recmul.asm, sort.asm, אותן ניתן כקלט לאסמבלר.

asm היא תוכנית האסמבלר – היא מקבלת כתובת לקובץ המכיל קוד בשפת אסמבלי ומתרגמת אותו לשפת מכונה (במקרה שלנו – פורמט של 4 ספרות הקסדצימליות).

על מנת להריץ את התוכנית יש להקליד בcmd . asm.exe program.asm mem.txt קובץ הקלט program.asm מכיל את תוכנית הAssembly וקובץ הפלט memin.txt מייצג relative path לקובץ בו נשמור את תוצאת התרגום של program.asm לשפת מכונה.

Sim היא תוכנית הסימולטור – היא מקבלת קובץ טקסט המכיל את תוכן הזיכרון הראשי בשפת מכונה, ומבצע את הפקודות המופיעות בו. על מנת להריצו, יש להקליד את השורה הבאה sim.exe memin.txt memout.txt regout.txt trace.txt count.txt.

הקובץ memin.txt הוא קובץ קלט בפורמט טקסט המכיל את תוכן הזיכרון הראשי בתחילת הריצה. הקובץ memout.txt הוא קובץ פלט שמכיל את תוכן הזיכרון הראשי בסוף הריצה. הקובץ regout.txt הוא קובץ פלט, שמכיל את תוכן הרגיסטרים R0-R15 בסוף הריצה. הקובץ trace.txt הוא קובץ פלט, המכיל שורת טקסט עבור כל הראה שבוצעה ע"י המעבד בפורמט pc inst R0-R15, כלומר קודם מופיע ה-program counter ואז הפקודה המבוצעת, ומיד אח"כ תוכן הרגיסטרים. הקובץ count.txt הוא קובץ פלט שמכיל את מספר הפקודות שבוצעו.

התוכנית addmat.asm – כותבת את סכום שתי מטריצות: הראשונה נמצאת בכתובות ox100-0x10f והשנייה ב0x110-ox11f למטריצה בגודל 4x4 הנמצאת בכתובות 0x120-0x12f.

התוכנית recmul.asm – המחשבת כפל של שני מספרים באופן רקורסיבי – האחד בכתובות ox100 והשני ב0x101 ואת התוצאה שמה בכתובת ox102.

התוכנית sort.asm – המבצעת מיון של 16 מספרים הנמצאים בכתובות ox100-0x10f בסדר עולה.

רעיונות האלגוריתמים של התוכניות

האסמבלר – האסמבלר קורא את קובץ הקלט שורה אחרי שורה (כל שורה היא פקודה), מוצא את כתובות הlabels ע"י שימוש ברשימה מקושרת) ושומר אותן. לאחר מכן הוא שוב עובר שורה שורה על הקובץ, כעת על הפקודות (קודם הרגילות ואז word.) טוען את מאפייני הפקודה (סוג הפקודה והפרמטרים) מתרגם אותם לערך אקסדצימלי (באורך 4 bytes). כיוון שהפרמטר האחרון בכל פקודה יכול להיות קבוע או label או opcode אז : אם זה opcode זה חלק מפקודת noimm, אם מדובר בlabel הוא מתורגם למספר השורה שמייצג הlabel ולאחר מכן לערך האקסדצימלי שמספר זה מייצג. הנתונים נכתבים במערך שמועתק לקובץ הפלט (memin).

הסימולטור- ליבו של הסימולטור בפונקציה הראשית "main". ה Program counter מתחיל בערך 0. בכל איטרציה קוראים את הפקודה הנוכחית לביצוע מהקובץ memin.txt, ומאחסנים אותה במערך דו ממדי MEM[MEM_SPACE][4] (כאשר MEM_SPACE=4096). כל שורה במערך הדו ממדי היא פקודה אחרת, וכל תא בכל שורה הוא ספרה אחת מבין ארבעת התווים (ההקסה- דצימליים) המרכיבים את אותה פקודה. בנוסף אתחלנו מערך של 16 אוגרים.

בכל איטרציה מדפיסים לקובץ trace.txt שורת טקסט המכילה את ערך האוגרים, ערך ה Program counter וההוראה המקודדת כפי שנקראה מ memin.txt.

עבור כל פקודה מבינים באיזה Opcode מדובר עפ"י האיבר הראשון במערך החד ממדי של אותה שורה הלוא הוא MEM[PC][0] ולאחר מכן מעדכנים את האוגר/ ה Program counter/שניהם בהתאם להנחיות של אותה פקודה.

ערך ה Program counter עולה באחד בכל איטרציה למעט עבור פקודות שמשנות את ערך ה Program counter ולכן הוא משתנה בהתאם אליהן.

נדגיש כי המעבר על שורות הקובץ memin.txt מתבצע לא בהכרח עפ"י סדר הופעתן בקובץ אלא על פי מיקום ה Program counter. כלומר, אם התקבלה פקודת קפיצה לדוגמא, ערך ה Program counter יתעדכן בהתאם ולכן הפקודה הבאה שתתבצע ע"י הסימולטור תהיה השורה המתאימה לערך ה Program counter המבוקש בקובץ memin.txt (ובפרט אינה השורה הבאה בקובץ).

הסימולטור מסיים את פעולתו בעזרת ביצוע פקודת halt. ברגע שנקראה פקודה זו, ראשית מדפיסים לקובץ regout.txt את תוכן האוגרים בסיום הריצה, לקובץ memout.txt את תוכן הזיכרון הראשי ולקובץ count.txt את מספר ההוראות שבוצעו ע"י הסימולטור, לאחר מכן מדפיסים למסך הודעה המבשרת על קבלת פקודת halt וסיום הריצה.