פרויקט – מעבד SIMP

מגישים: עדן ולנשטין 207134255

ענב שרגא 318513512

מור סער 208601351

אסמבלר

אסמבלר הוא כלי תכנות שמתרגם קוד בשפת אסמבלי לשפת מכונה. שפת מכונה היא השפה שהמעבד יכול להבין ולהריץ את התוכנית. לתוך האסמבלר נכנסים קבצי אסמבלי, עליהם נרחיב בהמשך. מתוך האסמבלר יוצא קובץ פלט שנקרא memin.txt, המכיל את הזיכרון הראשי לאחר המעבר לשפת מכונה.

הזיכרון מכיל לכל היותר 4096 שורות, כאשר כל שורה מקודדת לפי 5 ספרות הקסדצימליות. שני הספרות הראשונות בכל שורה מייצגים את הפעולה, ושלושת הספרות שאחריהם מייצגים את הרגיסטרים rd, rs, rt בהתאמה. אם באותה שורה היה שימוש בקבוע או בתווית(כלומר פעולת קפיצה), הקבוע או הכתובת של התווית יישמר בשורה מתחת לפקודה.

האסמבלר מבצע שני מעברים על קובץ הטקסט של התוכנית. במעבר הראשון, התוכנית סופרת לנו את מספר השורות שקיימים בקובץ האסמבלי. במעבר השני, האסמבלר מתרגם את השורות לשפת מכונה.

השתמשנו בספריות "ctype.h", "string.h", "stdlib.h", "stdio.h" ו – "stdbool.h".

המבנים שהוגדרו באסמבלר משמשים לארגון נתונים עבור האסמבלר:

Label- מכיל את שם התווית, מספר השורה בקוד האסמבלי, ומספר השורה בקוד המכונה. הוא משמש לשמירת כתובות של תוויות לצורך קפיצות בתוכנית.

Format- מכיל את הקוד הפקודה (opcode) רשימת הרגיסטרים (rd,rs,rt), ערך קבוע או תווית, ומספר השורה שבה נכתבה הפקודה בקוד המכונה. בנוסף, שמרנו האם כל פקודה שאנו פוגשים במעבר תקין. מבנה זה עוזר לתרגם את הפקודות לשפת מכונה.

RI_Format - מייצג את הפורמט של פקודה לאחר שתורגמה להקסדצימלי. הוא מכיל את קוד הפקודה וייצוג הקסדצימלי של הרגיסטרים והערכים. משמש לכתיבה סופית של הקוד.

pse - מייצג פקודת פסודו (.word) המכילה ערך קבוע, ומספר השורה בזיכרון שאליו נרצה להכניס את המספר. מבנה זה משמש לניהול והוספת נתונים לזיכרון.

<u>פונקציות עיקריות:</u>

open_fw ו- open_fr: פותחות קבצים לקריאה ולכתיבה בהתאמה.

count_lines_in_file: סופרת את מספר השורות בקובץ נתון.

create_pse_array ו- create_ri_format_array: יוצרות ומחזירות מערכים עבור פקודות פסודו ו- RI_Format בהתאמה.

free_pse_array ו- free_ri_format_array: משחררות את הזיכרון הדינאמי שהוקצה למערכים.

toLowerCase: ממירה מחרוזת לאותיות קטנות.

decoder_opcode ו- decoder_rrr: מפענחות את הקודים של הפקודות והרגיסטרים לפורמט הקסדצימלי.

write_to_text: כותבת את הפקודות בקובץ הפלט במבנה ההקסדצימלי, תוך טיפול בתוויות ובערכים קבועים.

find_label_new_line: מחפשת תווית במערך התוויות ומחזירה את מספר השורה המתאים בקוד המכונה.

print_to_file: כותבת את הפלט הסופי לקובץ, כולל עדכון פקודות פסודו.

write_memin: מעבד את הקוד האסמבלי, מזהה תוויות ופקודות פסודו, וממיר את הפקודות לשפת מכונה.

במעבר על הקוד, האסמבלר שומר במערכים שונים את כל המידע עבור כל שורה הכוללת פקודה ורגיסטרים. בנוסף, הוא שומר את מספר השורה שבה פעולה אמורה להתבצע בשפת המכונה. כאשר קיימת פקודה עם ערך קבוע (immediate) המכילה קבוע או תווית, האסמבלר מעדכן את ה-PC באותה תא במערך ב-2. לעומת זאת, אם אין ערך קבוע או תווית, ה-PC מתקדם ב-1.

לבסוף, לאחר מעבר על כל השורות ושמירה של הערכים עבור כל שורה במערך חדש, שבו כל איבר הוא מבנה מסוג RI_Format, נוכל ליצור את קובץ memin.txt על ידי הדפסה של כל תא בנפרד. במקביל, אם הייתה קיימת הוראת פסודו, האסמבלר שמר את כל ההוראות מסוג זה במערך, ובסיום תהליך ההמרה, כותב אותן לקובץ הזיכרון.

<u>סימולטור</u>

סימולטור הוא כלי תוכנה שמחקה את ההתנהגות של מערכת חומרה או מעבד, ומאפשר לבדוק ולנתח קוד בלי להשתמש בחומרה פיזית. הוא מבצע הוראות קוד מכונה ומספק פלטים כאילו הם רצים על החומרה האמיתית.

המבנים שהוגדרו בסימולטור:

Line_reg - מייצג שורת פקודה במעבד. זהו מבנה שמכיל את המידע הנדרש כדי לייצג הוראות עיבוד.

IO_reg - מייצג את הרגיסטרים של קלט/פלט (I/O) במערכת, כולל דגלים וסטטוסים שונים שקשורים לפעולה של המעבד.

<u>פונקציות עיקריות:</u>

open_fw, open_fr, open_fwyuv ו- open_fp: פותחות קבצים ל- קריאה, כתיבה, כתיבה בינארית ושינוי.

sign_extend_20_to_32: הרחבת ערך בן 20 סיביות למספר בן 32 סיביות, תוך שמירה על הסימן.

Init: קריאת ערכים מקובץ והמרה שלהם למערך של מספרים.

init_irq2: קריאת ערכים מהקובץ והמרה שלהם למערך irq2 של מספרים רגילים.

make_irq: בודק האם צריך לבצע פסיקה שקשורה לשעון או ל- irq2.

check_irq: בודק האם צריכה להתבצע פסיקה מכל הסוגים ובמידה וכן מודיע לסימולטור על ידי הדלקת המשתנה i_need_to_jump. בנוסף, מכבה או מדליק את הרגיסטרי קלט פלט המתאימים.

disk_RW: ניהול קריאות וכתיבות לדיסק בהתאם לפקודות מהרגיסטרים.

Simulator: דימוי הפעולה של מעבד באמצעות קריאת פקודות מקובץ הזיכרון, ביצוען, ועדכון רשומות IO. הפונקציה גם מנהלת את פעילות הדיסק, את ההפרעות ומביאה לסיום פעולות דימות שונות בהתאם לדרישות.

מהלך פעולה של הסימולטור:

ראשית, הגדרת משתנים שונים עבור קובץ קריאה, פרטי הוראה, ערכים זמניים, מדדי פריטים שונים כמו מונה מחזור, דגלי IRQ, והגדרת שטח זיכרון עבור המוניטור. הגדרת מערך io_registers עם ערכים של רגיסטרי IO ו IO_reg_names לשמות הרגיסטרים.

שנית, הסימולטור קורא כל שורה מקובץ הזיכרון (memin) ומבצע את ההוראות. בודק את מצב ההפרעות (IRQ) ומביא לתגובה אם נדרש. הסימולטור מבצע קפיצות בקוד אם נדרש על ידי שינוי ה PC על ידי הזזת המצביע לשורה המתאימה בקלט הזיכרון. הסימולטור מבצע את הפעולות הנדרשות ומעדכן בהתאם את הזיכרון וקורא לפונקציות שקשורות לדיסק במידת הצורך.

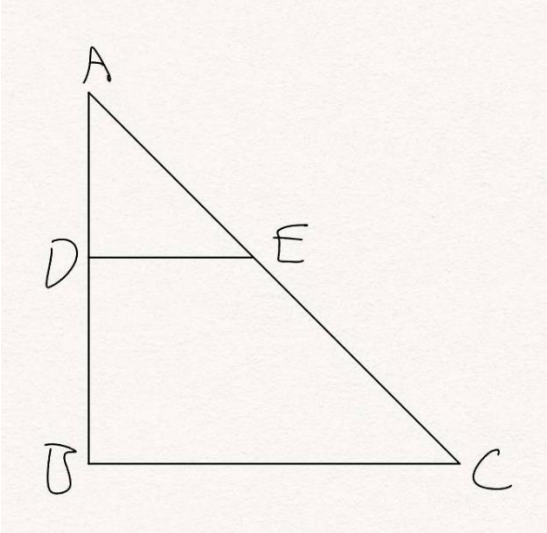
לאחר סיום ביצוע הפעולות ותיעוד שלהם בקובץ ה- trace ו-hwregtrace, הסימולטור מעדכן את הקבצים: כותב את המוניטור והזיכרון לקובץ monitor ו – monitor.yuv. כותב את מצב הלדים לתוך קובץ leds ו - display7seg. כותב את ערכי הרגיסטרים לקובץ regout. כותב את מספר המחזורים לקובץ cycles. כותב את תוכן הזיכרון לקובץ memout. כותב את תוכן הדיסק לקובץ diskout.

תוכניות בדיקה (קבצי האסמבלי)

Sort – מבצעת מיון של 16 מספרים בסדר עולה לפי מיון Bubble sort. המערך ההתחלתי של המספרים מופיעה בזיכרון (memin.txt) מכתובת 0x100 עד 0x10f. הקוד בוצע על ידי שתי לולאות for.

Binom - מחשבת את מקדם הבינום של ניוטון באופן רקורסיבי לפי אלגוריתם הידוע. בתחילת הריצה הארגומנט הראשון n נמצא בכתובת 0x100 והארגומנט השני k נמצא בכתובת 0x101. תוצאת ההרצה נכתבה לכתובת 0x102.

Triangle – הפונקציה מקבלת 3 קודקודים של משולש ישר זווית כאשר נק A נתונה בכתובת 0x100, נק B נתונה בכתובת 0x101 ונקודה C נתונה בכתובת 0x102. הנקודות מייצגות מיקום במוניטור והפונקציה משנה את ערכי הפיקסלים לצבע לבן בתוך המשולש ובהיקפו. במהלך הריצה כמות הפיקסלים שהודלקו חושבה לפי דמיון משולשים עם פונקציית חילוק שבוצעה בקוד האסמבלי כהרכבה של פעולות חיסור.



איור 1: משולש

$$\frac{DE}{BC} = \frac{AD}{AB} \rightarrow DE = AD * \frac{BC}{AB}$$

ראשית, חישבנו את המכפלה AD*BC ולאחר מכן חילקנו באורך AB לקבלת תוצאת מדויקת יותר. שאר הגדלים חושבו מפעולות חיבור חיסור שנעשו לפני.

*כל גודל מייצג כמות פיקסלים ותוצאות לא שלמות של החילוק עוגלו כלפי מטה.

Disktest - מבצע סיכום של תוכן סקטורים 0 עד 7 בדיסק הקשיח וכותבת את תוצאת הסיכום לסקטור מספר 8. החלטנו לא לאפשר פסיקות של הדיסק אלא לבדוק את סטטוס הדיסק דרך הקוד כאשר הקוד לא יבצע את פעולת כתיבה או קריאה הבאה מבלי שהדיסק סיים את הפעולה הקודמת.