פרויקט קורס מבנה המחשב

מגישים: גיא דוידי(205493448), רון מרק(208845685)

קורס מבנה המחשב

ASSEMBLER

סטראקטים (struct)

```
typedef struct label
    {
        char name[50];
        int line_num;
        struct label *next;
    }label;
```

הוגדר מבנה נתונים LABLE כמבנה נתונים רשימה מקושרת. מבנה נתונים זה שומר את כלל הלייבלים ואת מספר השורה בה הוא נמצא במעבר ראשוני של הקוד.

פונקציות:

1. new_label– במעבר הראשון על קוד האסמבלי לאחר זיהוי label מוסיף אותו לרשימה מקושרת ומעדכן את מספר השורה שלו.
2. label_to_line_num– מחפש האם ברשימה המקושרת של הlables קיים הlabel הרצוי, אם כן מחזירה את מספר השורה שלו
3. reg_sel – מקבלת טקסט המתאר את הרגיסטר הרצוי "$t0" למשל ומעדכנת את קוד המכונה (רצף 12 ספרות הקסה) במספר הרגיסטר המתאים
4. opcode_sel – מקבלת טקסט המתאר את הפעולה הרצויה "add" למשל ומעדכנת את קוד המכונה במספר ההוראה המתאים
5. imm_sel - מקבלת טקסט המתאר את האימידייט (immediate) הרצויה (label\decimal_number\hexa_number) ומעדכנת את קוד המכונה במספר המתאים, במידה וקיבלה לייבל היא קוראת ל label_to_line_num.
6. clean_list – בסוף הריצה מוחק את הרשימה המקושרת המכילה את הלייבלים ומשחרר את הזיכרון
7. word_func – פונקציה המטפלת בפרוצדורה המיוחדת word. משנה את קובץ dmemin במקום הרצוי לפני הסימולטור.

תהליך האסמבלר בקצרה:

מעבר ראשון על קוד האסמבלי – עובר על הקוד ומעדכן רשימת לייבלים עם ערכי השורות שלהם לשימוש עתידי

מעבר שני על קוד האסמבלי – עובר שורה שורה ומתרגם את הוראות האסמבלי לשפת מכונה תוך שימוש ברשימת הלייבלים שבנה במעבר הראשון.

את ההוראות שהוא מתרגם הוא כותב לקובץ imemin בתצורת 12 ספרות הקסדצימליות.

SIMULATOR

1. **סיפריות בסימולטור:**
   - stdio.h
   - stdint.h
   - stdlib.h
2. הוגדר union לצורך נוחות לגישה של בית ספציפי מתוך int:

```
union Data {
    unsigned i;
    unsigned char bytes[4];
} data;
```

3. הוגדרו קבועים לכלל הרגיסטרים המשומשים בסימולטור: רגיסטרי חומרה/מעבד.
4. הוגדרו קבועים

פונקציות:

1. My_shift – מימוש הזזת מספר בshift שמאלה פעמים.
2. countSetBits – פונקציה שסופרת את מספר הביטים הדלוקים במספר.
3. createMask – יוצרת מסכה לפי מספרי ביטים נתונים כלומר יצור מסכמה לפי דרישה בין ביט A לביט B.
4. init_list- פונקציה שמאתחלת רשימה לאפסים.
5. print_list – מדפיסה רשימה.
6. Myprintf -– פונקציית דיבאג – מדפיסה רשימה ללא האיבר של '\0'.
7. get_imemin_inst_from_file – מעבירה את קובץ הIMMIN לרשימה, הרשימה מתעדכנת לאורך התוכנית ובסוף נכתבת לקובץ הפרט dmemout.
8. zero_holder – מאפסת באפר של 4 char.
9. clear_interrupt – מאפסת את כל האינטרפטים שנדלקו לאחר סיום שגרת הפסיקה.
10. Monitor- פוקציה זו מקבלת את הממוניטור שמומש על ידי מטריצה של char (בית אחד) וכותבת את ה dataהרצוי אל הכתובת לפי דרישת המשתמש.
11. Read_write_Disk – פונקציה זו מחליטה לפי diskcmd האם לקרוא או לכתוב לדיסק הקשיח בסקטור ספציפי.
12. get_reg_name_by_table – מקבלת מספר רגיסטר חומרה ומחזירה string (מערך של char) של שם לפי טבלת הארכיטקטורה.
13. Execute – מקבלת את כלל ריגיסטרים לאחר שלב הפיענוח(decode) ומבצעת את הפקודת לפי טבלת הארכיטקטורה.
    פונקציה זו מפעילה בודקת האם אנו נמצאים בפסיקה ובודקת את הסיגנלirq.
    במידה וכן אנו נשלחים לשגרת הטיפול בפסיקה לפי הדרישה.
    אחרת אנו מבצעים את הפקודה לפי האופקוד שלה ובהתאם כותבים/ קוראים לזיכרון או לרגיסטר חומרה אחר.
    בסוף הפונקציה נחזיר ערך running המצביע על לולאת ה- fetch – decode execute להמשיך שלבים הפיענוח של הIMMIN.
    אם הגענו ל HALTנחזיר ערך 0 אשר יוציא אותנו מהתוכנית.
14. time_handler – מטפל הלוגיקה של הטיימר – מדליק בהגעה למספק השעון הרלוונטי את האינטרפט הנדרש.
15. Func – פונקציית דיבאג לבדיקת תקינות המסכה.

16. Mystrcpy – מעתיקה רשימה של char ללא התו של '\0'.
17. Creat_dmemout – יוצאת את קובץ הפלט dmemout.
18. Creat_regout – יוצאת את קובץ הפלט regout.
19. Creat_cycles - יוצאת את קובץ הפלט cycles.
20. get_irq2in_list – מעביר את ה data שנמצא בקובץ irq2 לרשימה על מנת לממש בדיקה נוחה של מספר השעון אל מול המידע בקובץ.
21. Is_value_in_list - בודק האם ערך נתו נמצא ברשימה הנתונה.
22. irq1_handler – בטמפל בלוגיקה של irq 1 – כלומר אם הערך ברשימה הנצונה מסעיף 20 נמצא אזי אנו נעבור לשיגרת הפסיקה בכך שנדליק את הביט ב.irqstatus
23. open_files – פונקציה שאחראית לפתוח את כל 14 הקבצים בסימולטור – מבדילה בעת הצורך בין הקבצים לדוגמה עבור קובץ YUV נקרא לצורך כתיבה בינארי ואחרים או קריאה או כתיבה.
24. close_files – פנקציה זו סוגרת את כל הקבצים בסוף התוכנית של הסיממולטור.
25. Decode – אחראית לקחת את הנצונים מתוך dmemin ולפרש אותם לנתונים מספריים ממשיים. הפיענוח עובר על כל הערכים של אופקוד, רגיסטרים אימידייתים (נשים לב שיש תמיכה כאן לקבלת מספר בsign extetion בתוך הimmediate).
26. HarDdisk_to_list – אחראי להעברת הDATA מתוך קוב הקלט של תוכן הסידק הקשיח למערך.
27. Creatmonitor – אחראית ליצור את קובץ הפלט monitor.txt. כותבת את המטריצה בקובץ טקסט בפורמט הרצוי.
28. Creatmonitoryuv – כותבת לקובץ בינארי על ידי המטריצה המייצגת את המוניטור.
29. Creatdiskout- מייצר את קובץ הפלט ל diskoutכלומר כותבת את המערך ששונה במהלך הריצה לקובץ טקסט.

MAIN

שלב א' יגדיר את מבני הנתונים בפרוייקט:

- מערך של קבצים לכלל הקבצים המתקבלים. FILE* Files[15]
- מערך לנתונים בקובץ IRQ2- int irq2_list[LEN_OF_RAM]
- interrupt_mode יגידיר האם אנו בריצה במצב של פסיקה.
- מערך רגיסטרי המעבד - simp_reg[LEN_OF_SIMP_REG]
- מערך רגיסטרי החומרה- IORegister[LEN_OF_HW_REG]
- מערך המחזיקה את נתונים הDMEMIN dmemin_list_RAM[LEN_OF_RAM]
- מטריצה המחזיקה את הוראות המעבד לריצה imemin_inst[LEN_OF_RAM][LEN_48_BITS]
- מערך המחזיק את הנתונים בדיסק הקשיח-disk_list[2048]
- program_counter יאותחל לאפס וירוץ בתוך לולאה הסמסלצת את ה fetch – decode -execute.
- יאותחלו כלל המשתנים לפני הפעינוח של פונק הdecode.

שלב ב' לולאת הfetch – decode -execute

1. Fetch- תחילה ניקח את ההוראה מהרשימה שהגדרנו מתוך קובץ הקלט.
2. DECODE – נשלח לתוך פונקציה שמפענחת את כלל הרגיסטרים והנתונים הנדרשים לביצוע ההוראה.
3. Execute– נשלח את כל מבני הנתונים עם ההוראה והנתונים הנדרשים לביצוע ההוראה.

שלב ג' יצירת קבצי הפלט.

שלב ד' סיום התוכנית.