**דוקומנטציה חיצונית לפרויקט במבנה המחשב**

**הסימולטור:**

**מבני הנתונים:**

1. רשימה מקושרת לשמירת שורות trace בשם trace\_line\_list.
2. רשימה מקושרת לשמירת שורות hwtrace בשם hw\_trace\_line\_list.
3. רשימה מקושרת לשמירת שורות לקובץ ה-Led בשם leds\_trace\_list.
4. רשימה מקושרת לשמירת שורות לקובץ dis7seg בשם dis7seg\_trace\_list.
5. רשימה מקושרת לשמירת אינטרפטי irq2 בשם irq2\_list.
6. מטריצה 256X256 לשמירת נתוני המוניטור בשם monitor.
7. מערך של instruction (struct המכיל שדות של חלקי הפקודה) באורך 4096 לשמירת כל ה-instructions שמופיעים בקובץ imemin.txt, בשם imemin\_instructions\_array.
8. מערך באורך 4096 לשמירת הזכרון בשם dmem\_array.
9. מטריצה 128X128 לשמירת נתוני הדיסק בשם disk\_matrix.
10. מערך פוינטרים לרגיסטרים בשם reg\_pointer\_array.
11. מערך פוינטרים לרגיסטרי החומרה בשם IO\_reg\_pointer\_array.

**משתנים חשובים:**

1. Cycle\_counter שעוקב אחר הסייקלים.
2. משתנה לכל רגיסטר ולכל רגיסטר-חומרה.
3. PC.
4. מעקב אחר disk\_timeout.
5. מעקב אחר סיגנל נוכחי curr\_sig.
6. משתנה עבור ה-node הנוכחי בכל רשימה מקושרת.

**פונקציות:**

1. **פונקציות לפרסור קבצי קלט**
   1. Init\_instructions\_array
      1. קוראת מקובץ ה-imemin.txt (או שם אחר) ומאתחלת את המערך imemin\_instructions\_array.
   2. Init\_dmem\_array
      1. קוראת מקובץ ה-dmem.txt ומאתחלת את מערך dmem\_array.
   3. Load\_diskin
      1. קוראת מקובץ ה-diskin.txt, ומאתחלת את מטריצת disk\_matrix.
   4. Load\_irq2in
      1. קוראת מקובץ ה-irq2in.txt, ומאתחלת את הרשימה irq2\_list.
      2. פונקציית עזר ליצירת תא יחיד ברשימה והוספתו- add\_irq2\_node. מקבלת כארגומנט את מספר הסייקל.
   5. Load\_files
      1. קוראת לכל יתר הפונקציות בסקשיין. אם הייתה שגיאה, יוצאת מהריצה.
2. **פונקציות ליצירת קבצי פלט**
   1. Open\_w\_then\_a, open\_in\_mode
      1. זוג פונקציות שמטרתן קלות בפתיחת קובץ חדש, באופן כזה שירסט את תוכנו אם היה קיים- ויאפשר להוסיף לו תוכן.
   2. Convert\_instruction\_to\_bits
      1. מקבלת struct של instruction, מחזירה את רצף הביטים המייצג אותו.
   3. Find\_limit\_index\_in\_dmem, find\_limit\_index\_in\_diskout, find\_limit\_index\_in\_monitor
      1. פונקציות שמטרתן מציאת האינדקס שהחל ממנו הזכרון, הדיסק והמוניטור הם אפסים בלבד (כדי לא לכלול אותם בקובץ הפלט).
   4. Create\_dmemout\_txt
      1. יצירת קובץ ה-dmemout.txt מתוך dmem\_array.
   5. Create\_regout\_txt
      1. יצירת קובץ ה-regout.txt מתוך ערכי הרגיסטרים בסיום.
   6. Create\_trace\_txt
      1. יצירת קובץ ה-trace.txt מתוך הרשימה trace\_line\_list.
   7. Create\_hwregtrace\_txt
      1. יצירת קובץ ה-hwregtrace.txt מתוך הרשימה hw\_trace\_line\_list.
   8. Create\_cycles\_txt
      1. יצירת קובץ ה-cycles.txt מתוך ערך המשתנה cycles\_counter.
   9. Create\_leds\_txt
      1. יצירת קובץ ה-leds.txt מתוך הרשימה leds\_trace\_list.
   10. Create\_display7seg\_txt
       1. יצירת קובץ ה-display7seg.txt מתוך הרשימה display7seg\_list.
   11. Create\_diskout\_txt
       1. יצירת קובץ ה-diskout.txt מתוך המטריצה disk\_matrix.
   12. Create\_monitor\_txt
       1. יצירת קובץ ה-monitor.txt מתוך המטריצה monitor.
       2. יצירת קובץ ה-monitor.yuv מתוך המטריצה monitor.
   13. Create\_out\_files
       1. קוראת לכל פונקציות יצירת הקבצים.
3. **פונקציות לניהול ומעקב הממשק עם הIO**
   1. Handle\_cmds
      1. מבצעת בדיקה האם פקודת ה-out כתבה פקודה לדיסק או למוניטור, ומדמה את הביצוע של הפקודה בהתאם.
   2. Io\_commands\_trace
      1. מבצעת מעקב אחר פעולות ה-IO באמצעות הרשימות המקושרות הרלוונטיות.
   3. Add\_hw\_trace\_node
      1. יוצרת ומוסיפה תא חדש לרשימה המקושרת hw\_trace\_line\_list.
   4. Add\_leds\_trace\_node
      1. יוצרת ומוסיפה תא חדש לרשימה המקושרת leds\_trace\_list.
   5. Add\_dis7seg\_trace\_node
      1. יוצרת ומוסיפה תא חדש לרשימה המקושרת display7seg\_list.
4. **פונקציות המממשות instructions**
   1. Do\_instruction\_command(relevant registers) הוא המבנה הכללי, כאשר instruction זה פלייס הולדר לכל אחת מהפקודות המופיעות בטבלה. מקבלות כארגומנטים את מספרי הרגיסטרים הרלוונטיים לביצוען.
   2. Commit\_the\_instruction(instruction)
      1. מקבלת struct של instruction, וקוראת לפונקציה המתאימה לביצועה. מחזירה 1 אם הצליחה, 1- אם קרתה שגיאה, ו-0 אם הפקודה היא Halt.
5. **פונקציות עזר ל-main**
   1. Add\_trace\_node
      1. יצירה והוספת תא לרשימה המקושרת trace\_node\_list.
   2. Copy\_regs\_array
      1. מחזירה מערך של סנפשוט הרגיסטרים באותה נקודת זמן.
   3. Set\_up\_files
      1. מפרסרת ושומרת את כל ה-Pathים של הקבצים משורת הריצה.
   4. Handle\_ints
      1. מתמודדת עם אינטרפטים, מעדכנת את רגיסטרי ה-IO בהתאם.
   5. Exec\_instruction
      1. מריצה את commit\_the\_instruction. מסיימת את הריצה אם זו החזירה שגיאה. מעדכנת את cycle\_counter, pc. מחזירה האם הפקודה היא halt.
   6. Update\_immediates
      1. מעדכנת את הרגיסטרים של Imm1, imm2 לערכים שניתנו ב-instruction.
6. **Main**
   1. מטפלת בארגמונטי ההרצה
   2. טוענת את קבצי הקלט
   3. מבצעת לולאה של סייקל עד קבלת Halt \ שגיאה.
   4. יוצרת את קבצי הפלט

**פלואו של סייקל בודד:**

1. מעקב והתמודדות עם אינטרפטים.
2. עדכון הרגיסטרים immediates.
3. הוספת תא עבור trace.
4. ביצוע הפקודה שניתנה (ועדכון כל יתר המשתנים ומבני הנתונים הנגזרים מכך).

**האסמבלר:**

1. **מבני הנתונים**
2. **משתנים חשובים**
3. **פונקציות**
4. **פלואו כללי**

**הטסטים:**

**Multmat.asm**

1. שימוש:
   1. ביצוע כפל מטריצות 4X4 מהזיכרון המדומה, ושמירת התוצאה בזיכרון המדומה.
2. מבנה:
   1. פונקציית main הכוללת שתי לולאות (LOOP1\_IN\_MAIN, LOOP2\_IN\_MAIN).
   2. פונקציית CREATE\_CELL הנקראת בכל איטרציה, המחשבת את הערך בכל תא במטריצת התוצאה. מכילה את הלולאה LOOP\_IN\_CELL.
   3. בסוף יש רצף שורות אתחול של זיכרון, לא כחלק מהקוד עצמו- אלא כדי ליצר תהליך שאינו הכפלת מטריצת 0 במטריצת 0.
3. פסאודו קוד:

Def main():

For(int i=0; i<4;i++):

For (int j=0;j<4;j++):

Create\_cell(i,j)

Def create\_cell(int i, int j):

Int tmp = 0

For(int k=0;k<4;k++):

Tmp = tmp + matA[i][k]\*matB[k][j]

matC[i][j] = tmp

* 1. $a0 -> i, $a1 -> j, $t0 -> k, $t1 -> tmp
  2. אין הזזה בקוד הנ"ל של $sp, מאחר שלא בוצע שימוש ברגיסטרים שדורש זאת.

**Binom.asm**

1. שימוש:
   1. מחשב את התוצאה של עבור n, k חיוביים.
2. מבנה:
   1. טעינת ערכים מהזיכרון וכניסה ללולאה המרכזית.
   2. לולאה מרכזית - חלוקה למקרי קצה ולולאה כללית וקפיצה בהתאם.
      1. מקרי קצה – מגדילים את התוצאה ב-1.
      2. מקרים כלליים – מאחסנים את כל הפרמטרים ואת כתובת החזרה ב-stack, וקוראים ללולאה הכללית פעמיים – פעם אחת עם n – 1, k – 1 ופעם אחת עם n – 1, k.
   3. שמירה מראש של ערכים לזכרון

**Circle.asm**

1. שימוש: יצירת מעגל בצבע לבן ברדיוס הנדרש.
2. מבנה:
   1. טעינת הרדיוס מהזיכרון ושמירת הרדיוס בריבוע
   2. לולאה חיצונית –
      1. איתחול קוארדינטת ה-X ל-0
      2. חישוב המרחק הנוכחי בציר ה-Y בריבוע
   3. לולאה פנימית –
      1. חישוב המרחק הנוכחי בציר ה-X בריבוע
      2. שמירת סכום המרחקים בריבוע (כל אחד בריבוע בנפרד)
      3. בדיקה אל מול הרדיוס בריבוע
      4. צביעה בלבן במקרה והמרחק מהמרכז (128, 128) קטן או שווה לרדיוס
   4. סוף הלולאה –
      1. העלאת קואורדינטת ה-X כל פעם וקפיצה ללולאה הפנימית
      2. בכל 256 ריצות של הלולאה הפנימית, העלאת קואורדינטת ה-Y באחד וקפיצה ללולאה החיצונית.

**Disktest.asm**

1. שימוש:
   1. העתקת תוכן הסקטורים 0-7 לסקטורים 1-8 בהתאמה. (לא "מחקנו" את התוכן של סקטור 0, מאחר שלא ראינו דרישה לכך- אלא רק דרישה להמצאות תוכן ב1-8).
2. מבנה:
   1. פונקציית main הכוללת לולאה LOOP.
   2. פונקציית MOVE\_SECTOR הכוללת שתי לולאות (WHILE1, WHILE2) הדואגת לקרוא את תוכן סקטור מסוים, ולכתוב אותו לסקטור העוקב.
3. פסאודו קוד:

Def main():

Diskbuffer = 0;

For(int i=7; i>-1; i--):

Move\_sector(i);

Def move\_sector(i):

While (disk\_not\_ready){};

Disksector = i;

diskcmd = read;

While (disk\_not\_ready){};

Disksector = i+1;

Diskcmd = write;

* 1. $a0 -> i
  2. הבאפר נותר קבוע מאחר שבכל איטרציה הסקטור נקרא ואז נכתב, והתוכן נדרס באיטרציה הבאה (כי הוא כבר בדיסק).
  3. אין הזזה בקוד הנ"ל של $sp, מאחר שלא בוצע שימוש ברגיסטרים שדורש זאת.