**מבוא**

המסמך מתאר את כל מרכיבי הקוד של האסמבלר והסימולטור לפרויקט SIMP. הפרויקט נועד לאפשר הידור והרצה של קוד בשפת אסמבלי המיועד למעבד RISC פשוט בשם SIMP. המסמך מסביר כל פונקציה, מבנה נתונים ומנגנון בקוד בצורה מפורטת.

**אסמבלר**

**תיאור כללי**

האסמבלר ממיר קוד אסמבלי לקוד מכונה המותאם לזיכרון הפקודות ולזיכרון הנתונים של מעבד SIMP. התהליך כולל שתי מעברים:

1. **מעבר ראשון**: זיהוי תוויות והקצאת כתובות זיכרון לכל תווית.
2. **מעבר שני**: קידוד פקודות אסמבלי לפקודות בינאריות והכנת קובצי פלט.

**מבני נתונים עיקריים**

* **LineInfo**: מבנה שמייצג שורת קוד אסמבלי וכולל את הטקסט, האם זו פקודה, ומה הכתובת שלה.
* **Label**: מבנה שמייצג תווית, כולל את שמה ואת הכתובת שהיא מייצגת בזיכרון.
* **dmem**: מערך לזיכרון נתונים בגודל המקסימלי של זיכרון הנתונים.

**פונקציות עיקריות**

* **opcode\_of**: מחזירה את מספר האופקוד המתאים לשם האופקוד.
* **register\_number**: ממירה שם של רגיסטר למספר שלו.
* **find\_label\_address**: מחפשת את כתובת הזיכרון המיוצגת על ידי תווית.
* **parse\_immediate\_final**: ממירה ערך מידי למספר מתאים. אם מדובר בתווית, היא תמצא את כתובתה.
* **encode\_instruction**: מקודדת פקודה לפורמט בינארי של ארבעים ושמונה ביטים.

**מעבר ראשון**

מעבר זה סורק את הקובץ ומאתר תוויות. כל תווית מקבלת כתובת בזיכרון, והאסמבלר שומר אותה במבנה הנתונים של התוויות.

**מעבר שני**

מעבר זה מתמקד בקידוד הפקודות:

* פקודות קידוד נכתבות לקובץ זיכרון הפקודות.
* פקודות מסוג ".word" מעדכנות את זיכרון הנתונים.
* המידע נכתב לקובצי הפלט במבנה מתאים.

**סימולטור**

**תיאור כללי**

הסימולטור מדמה את ביצוע פקודות המעבד בפועל. הוא פועל על ידי ביצוע שלבי "שאיבה-פענוח-ביצוע" לכל פקודה בזיכרון הפקודות. הסימולטור מנהל גם את רכיבי הקלט/פלט של המעבד, כולל שעון, דיסק, צג ותאורת לד.

**מבני נתונים עיקריים**

* **imem**: מערך עבור זיכרון הפקודות.
* **dmem**: מערך עבור זיכרון הנתונים.
* **disk**: מערך עבור הדיסק המדומה.
* **monitor**: מערך המייצג את זיכרון הצג.

**פונקציות עיקריות**

* **read\_imem**: טוענת את זיכרון הפקודות מקובץ.
* **read\_dmem**: טוענת את זיכרון הנתונים מקובץ.
* **read\_disk**: טוענת את תוכן הדיסק מקובץ.
* **read\_irq2**: טוענת את מועדי הפעלת הפרעות חיצוניות מקובץ.
* **decode\_instruction**: מפענחת פקודה ומוציאה את רכיביה.
* **log\_hwregtrace**: רושמת פעולות קריאה/כתיבה ברשומות חומרה.
* **check\_leds**: בודקת שינויים בתאורת הלד ומעדכנת את קובץ הפלט.
* **check\_7seg**: בודקת שינויים בתצוגת שבעה מקטעים ומעדכנת את קובץ הפלט.
* **check\_irq2**: מזהה מתי יש להפעיל את ההפרעה החיצונית.
* **update\_timer**: מעדכנת את השעון ומפעילה את ההפרעה המתאימה.
* **update\_disk**: בודקת ומעדכנת מצב פעולות דיסק.
* **check\_interrupts**: בודקת אם יש הפרעות ופועלת בהתאם.
* **write\_monitor\_pixel**: כותבת ערכים לזיכרון הצג.
* **print\_trace\_line**: רושמת את מצב הפקודה הנוכחית לקובץ המעקב.
* **execute\_instruction**: מבצעת את הפקודה הנוכחית במעבד המדומה.

**ניהול קלט/פלט**

הסימולטור מטפל ברשומות הקלט/פלט באמצעות כתיבה וקריאה ממערך רשומות החומרה. כל שינוי נרשם לקובצי הפלט המתאימים.

**מעגל ביצוע**

הסימולטור פועל בלולאה המחולקת לשלבים:

1. עדכון שעון החומרה.
2. בדיקת הפרעות חיצוניות.
3. עדכון שעון פנימי.
4. בדיקת מצב הדיסק.
5. בדיקת הפרעות.
6. ביצוע פקודה.
7. ספירת מחזורי שעון.

**קובצי פלט**

בסיום ההרצה, הסימולטור כותב את מצב הזיכרון, הרגיסטרים, והחומרה לקובצי הפלט הנדרשים.

**מסקנות וסיכום**

הקוד מתמקד בהמרת קוד אסמבלי לקוד מכונה והרצת סימולציה מדויקת של מעבד SIMP. באמצעות שתי יחידות נפרדות - האסמבלר והסימולטור - ניתן לכתוב, להדר ולהריץ תוכניות המותאמות לארכיטקטורת המעבד.

המבנה המודולרי של הקוד והטיפול המדויק בזיכרונות וברכיבי חומרה מבטיחים התאמה מלאה לדרישות הפרויקט.

**תוכנית חישוב מקדם בינומי**

**מטרת התוכנית**

התוכנית מחשבת את המקדם הבינומי עבור ערכים נתונים של משתנים באמצעות חישוב רקורסיבי. הפלט של החישוב נשמר בזיכרון בכתובת המתאימה.

**שלבי התוכנית**

* הגדרת נקודת סיום התוכנית באמצעות כתובת קפיצה.
* אתחול מצביע המחסנית עם כתובת התחלה גבוהה.
* טעינת הערך הראשוני של המשתנים מכתובות זיכרון.
* אתחול רקורסיבי של חישוב המקדם באמצעות קריאות חוזרות לפונקציה הראשית.
* התוכנית מבצעת השוואות בין המשתנים ומנתבת את זרימת הפקודות לפי תנאים מתאימים.
* בכל חזרה מהפונקציה, הערכים מסוכמים במשתנה המיועד לתוצאה.

**שימוש ברגיסטרים**

* רגיסטר ייעודי לתוצאה הסופית.
* רגיסטרים לכתובות הזיכרון של המשתנים.
* מצביע מחסנית לשמירת הערכים והחזרה לפונקציה הקוראת.

**תוכנית ציור מעגל**

**מטרת התוכנית**

התוכנית מציירת מעגל מונוכרומטי על מסך בגודל קבוע, עם רדיוס שמוגדר בזיכרון. המעגל ממורכז על המסך, וכל הפיקסלים שבתוך המעגל מקבלים ערך לבן.

**שלבי התוכנית**

* טעינת רדיוס המעגל מכתובת זיכרון.
* חישוב הערך הריבועי של הרדיוס ושמירתו ברגיסטר.
* התחלת לולאה לשינוי ערכי צירי המסך תוך בדיקת פיקסלים בתוך תחום המעגל.
* לכל פיקסל בתוך התחום מחושבת כתובת הזיכרון המתאימה, והוא מתעדכן לערך הלבן.
* ניהול לולאות פנימיות לכל ערכי ציר המסך.

**שימוש ברגיסטרים**

* רגיסטר לערך הרדיוס.
* רגיסטרים לערכים ריבועיים של קואורדינטות המסך.
* רגיסטר לכתובת הזיכרון של כל פיקסל.

**תוכנית כפל מטריצות**

**מטרת התוכנית**

התוכנית מבצעת כפל בין שתי מטריצות בגודל קבוע ושומרת את תוצאת הכפל במטריצה שלישית בזיכרון.

**שלבי התוכנית**

* אתחול כתובות הבסיס של המטריצות בזיכרון.
* התחלת לולאות חיצוניות ופנימיות לחישוב ערכי כל תא במטריצה התוצאה.
* עבור כל תא, מחושב סכום המכפלות של הערכים המתאימים מהשורות והעמודות.
* שמירת הערך המחושב בתא המתאים במטריצה השלישית.
* לולאות פנימיות מנהלות את התקדמות האינדקסים של השורות והעמודות.

**שימוש ברגיסטרים**

* רגיסטרים לכתובות הבסיס של המטריצות.
* רגיסטרים לערכי אינדקסים של שורות, עמודות ותאים.
* רגיסטרים לערכים זמניים עבור מכפלות וסכומים.

**תוכנית בדיקת דיסק**

**מטרת התוכנית**

התוכנית מדמה תהליכי קריאה וכתיבה מדיסק תוך שימוש במנגנוני DMA והפסקות חומרה. היא מעתיקה תוכן ממקטע אחד במכשיר הדיסק למקטע אחר.

**שלבי התוכנית**

* הפעלת רשומת החומרה המנהלת את הפסקות הדיסק.
* בדיקת מצב הדיסק כדי לוודא שהוא מוכן לביצוע פעולה.
* התחלת לולאות קריאה וכתיבה עבור מקטעי הדיסק.
* המתנה למחזורי זמן מתאימים בין פעולות הקריאה והכתיבה.
* סיום התוכנית כאשר כל המקטעים הרצויים הועתקו.

**שימוש ברגיסטרים**

* רגיסטרים לניהול כתובות המקטעים בדיסק.
* רגיסטרים למצב הדיסק ולמצביעי זיכרון.
* רגיסטרים לערכים זמניים לניהול לולאות.

**מסקנות וסיכום**

התוכניות האסמבליות שפותחו מממשות תהליכים מורכבים בצורה מובנית, יעילה ומותאמת לארכיטקטורת SIMP. כל תוכנית משתמשת ברגיסטרים ובזיכרון בצורה אופטימלית תוך הקפדה על ניהול זרימת הפקודות והפקת תוצאות מדויקות. המסמך מספק תיעוד מלא וברור של הקוד, ומבטיח הבנה מלאה של התהליכים עבור כל משתמש או מתכנת שיידרש לעבודה על מערכת זו בעתיד.