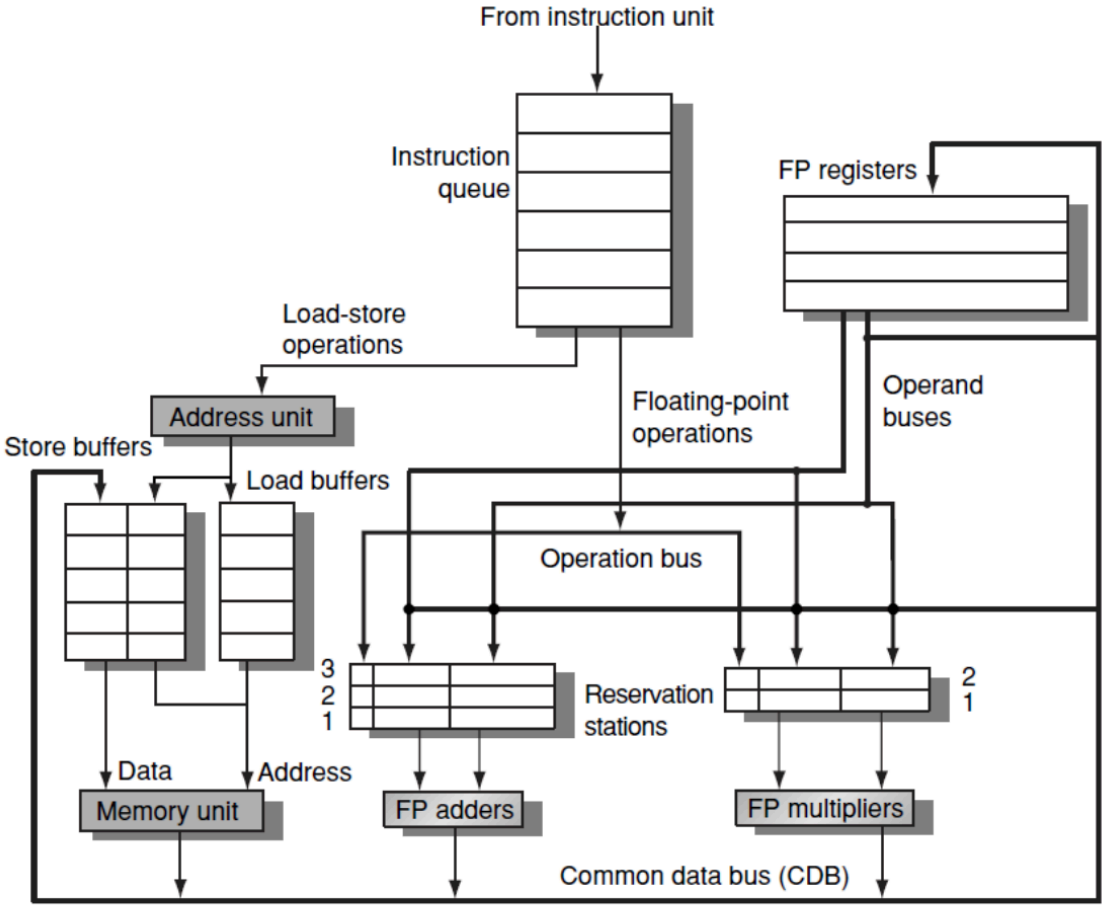
מימוש סימולטור למעבד FLOATING POINT המשתמש באלגוריתם Tomasulo

בפרוייקט זה מימשנו סימולוטור למעבד FLOATING POINT המשתמש באלגוריתם Tomasulo בשפת C.



קבצי המקור אשר מרכיבים את הסימולטור

- Memory – מערך אשר מייצג את אובייקט הזיכרון, כל פקודות הגישה לזיכרון ממומשות בקובץ המקור הזה. אתחול לזיכרון, קריאה וכתיבה לזיכרון, כתיבה לקובץ memout.
- Register – אובייקט המייצג רגיסטר, כולל ערך, תג ודגל האם הערך תקין.
- Instruction – אובייקט המייצג הוראה בודדת. אובייקט זה מחזיק שדות של ההוראה עצמה וכן של כל מרכיביה(OPCODE, רגיסטרים אינדקס ההוראה). בנוסף לכך, מוחזקים לכל הוראה כלל הזמנים הרלוונטים להוראה(issur cycle, execution cycle, writeCDB cycle ועוד).

- InstructionQueue – אובייקט המייצג את תור ההוראות. באובייקט זה קיימים שתי רשימות מקושרות. האחת להוראות אשר עברו את שלב issue, והאחרת עבור אלה שלא. האובייקטים המרכיבים רשימות אלו הן ה-Instruction
- ReservationStation – אובייקט זה מייצד תחנת המתנה בודדת. במידה והתחנה פעילה(דגל busy דלוק), היא מחזיקה את הרכיבים הרלוונטים לביצוע ההוראה כמו אינדקס ההוראה, opcode, ערכי הרגיסטרים הרלוונטים(או התגים שלהם). במידה וכלל רכיבי ההוראה קיימים, דגל הready נדלק וכאשר תהיה יחידה פונקציונלית מתאימה – ההוראה תוכל להתבצע.
- ReservationStationTable – מחלקה זו מחזיקה כ5 מערכים של ReservationStation (אחד עבור כל סוג של תחנה, לדוגמא mul,add). בנוסף היא מחזיקה עבור כל מערך את סך התחנות הקיימות במערך וכן את סך התחנות אשר בשימוש בכל רגע נתון. באמצעות אובייקט זה ניתן לגשת ביתר קלות לכלל ה-ReservationStation על פי סדר קבוע על מנת לבצע פעולות מסויימות(כמו כתיבה לCDB)
- Main – הפונקיה הראשית שלנו. הפונקציה יוצרת אובייקט CPU על פי הארגומנטים הניתנים לתוכנית. לאחר מכן מקבלת פקודת runCPU עד אשר נגיע לסיום הרצת הסימולטור ונבצע פקודות סיום לשחרור הזיכרון.
- CPU – האובייקט הראשי איתו נעבוד, מחזיקה אובייקטים של תור ההוראות, תחנות ההמתנה, רגיסטרים, זיכרון. בנוסף, אובייקט זה מכיל את קונפיגורציית המעבד(מס' יחידות פונקציונליות, מס' יחידות המתנה, השהיות, מספר יחידות בשימוש וכו'). בכל מחזור שעון אחראית על ביצוע כלל הפעולות של המעבד מול האובייקטים השונים(קריאת הוראות, ביצוע issue, חישוב התוצאה וכו')

אופן פעולת הסימולטור

- התוכנית מקבלת כקלט בשורת הפקודה: קובץ קונפיגורציה של המעבד, תמונת זיכרון אשר מכילה את ההוראות לביצוע, וכן נתיבים לקבצי הפלט שלנו.
- התוכנית תאתחל את מעבד(אובייקט CPU על פי קובץ הקונפיגורציה)
- אובייקט זה יאתחל את טבלת תחנות ההמתנה, את תור ההוראות ואת הרגיסטרים.
- התוכנית תתחיל את ריצתה (פונקציית runCPU)
- תחילה, התוכנית תייבא את הזיכרון לאובייקט הזיכרון
- התוכנית תקרא 2 הוראות ותוסיפם לתור ההוראות שלא בוצע להן issue
- לאחר מכן, בלולאה, כאשר כל איטרציה מהווה מחזור שעון בודד של המעבד נבצע:
  - issue ל2 הוראות(בתנאי שקיימות) – המעבד לוקח את 2 ההוראות הבאות מהתור של ההוראות שלא בוצע עבורן issue ומחפש עבורן תחנת המתנה פנויה, במידה וקיימת, מכניס את הוראת לתחנת ההמתנה הייעודית.
  - קריאת 2 ההוראות הבאות מהזיכרון לתור של ההוראות שלא בוצע עבורן issue
  - Execute – במידה ויש תחנות פונקציונליות פנויות וכן קיימות הוראות בתחנות המתנה הממתינות לביצוע(דגל ready). נעביר את ההוראות הממתינות לביצוע ולבסוף נכתוב את תוצאתם על ה-CDB(במחזור השעון הרלוונטי)
  - כתיבת התוצאות – בשלב זה אנחנו מדמים את הכתיבה על ה-CDB. תוך התחשבות בכך שיש לנו 4 CDB בסה"כ.

- כאשר נגיע להוראת halt וכל ההוראות בתחנות ההמתנה מסיימות לבצע נסיים את ריצת הלולאה
- נכתוב את קבצי הפלט
- נהרוס את האובייקטים ושחרר את הזיכרון

הנחות

- יתבצע issue להוראה לפחות מחזור שעון אחד אחרי שתקרא מהזיכרון
- הוראה תתחיל ביצוע לפחות מחזור שעון אחד אחרי שעשינו לה issue
- יש זיכרון יחיד
- חייבת להיות הוראת HALT אשר תעצור את הסימולטור
- קבצי הקלט הינם בפורמט אשר ניתן לנו
- שחרור היחידה הפונקציונלית עבור פעולת STORE תעשה מחזור שעון אחד לאחר סיום שלב הexecute
- עבור פעולת STORE – נגדיר את ערך cycle_write_cdb=-1 בקובץ ה-traceinst.txt (זאת משום שאין כתיבה לcdb עבור פעולה זו)

דוגמאות הרצה

- בדיקת זיכרון – טעינה מהזיכרון, ביצוע פעולה אריתמטית וכתיבה לזיכרון
- בדיקת מספר פעולות זיכרון ופעולת חילוק – טענו 2 ערכים מהזיכרון, חילקנו את הראשון מביניהם ב2 וכתבנו לזיכרון את התוצאה, תוך שימוש ביחידה פונקציונלית אחת של Load
- בדיקת זיכרון שניה, כתיבה לזיכרון ואז טעינה מהזיכרון(מאותה הכתובת)