**פרוייקט Tomasulo**

**מבנה הנתונים**

**Package: main**

Parser

מטרת ה- Parser היא לפרסר את קבצי ה- input בתחילת התוכנית וליצור את קבצי ה- output בסופה.

Memory

Class זה מייצג את זיכרון התוכנית. מכיל מערך בגודל 1024 של integers המייצג את הזיכרון. בנוסף מכיל מתודות של load / store עבור פקודות הדורשות גישה לזיכרון.

Instruction

מייצג פקודה בתוכנית. מכיל את השדות הבאים:

* שדות הפקודה – OPCODE, DST, SRC0, SRC1, IMM.
* מספר המחזור שבו התבצע שלב מסויים של הפקודה: issue\_cycle, execute\_start\_cycle, execute\_end\_cycle, write\_to\_CDB\_cycle.
* התוצאה של הפקודה – result. למשל עבור add, כאשר הפקודה תסיים את שלב ה- execute, שדה ה- result יתעדכן עם התוצאה של ה- add. יכול להיות int או float.
* binaryInst – הפקודה בייצוג בינארי.
* Station – ה- reservation station / buffer שבו הכנסנו את הפקודה (לא רלוונטי לפקודות מסוג branch או jump).
* PC – ה- PC של הפקודה.

ל- class זה מספר פונקציות תומכות:

* freeStation – כאשר הפקודה הסתיימה וביצענו כתיבה ל- CDB, מתודה זו תשחרר את ה- reservation station / buffer הרלוונטי לפקודה ותהפוך אותו ל- Not busy.
* isReadyToBeExecuted – מחזיר TRUE כאשר הפקודה מוכנה לעבור משלב ה- issue לשלב ה- execute.

Global

מחזיק מספר קבועים סטטיים של התוכנית.

Tomasulo

זהו ה- flow של התוכנית, יפורט בהמשך.

Package: buffers

Buffers

Class זה מייצג את כל ה- load & store buffers שיש למעבד. באתחול מקבל את מספר הבאפרים מכל סוג על פי קובץ הקונפיגורציה של המעבד ומאתחל לפי זה את הבאפרים. פונקציות תומכות:

* getFreeLoadBuffer – מחזיר load buffer שאינו busy. במתודה זו נשתמש בשלב ה- issue.
* getFreeStoreBuffer – מחזיר store buffer שאינו busy. במתודה זו נשתמש בשלב ה- issue.
* isThereFreeLoadBuffer – מחזיר true אם יש load buffer פנוי. במתודה זו נשתמש בשלב ה- issue.
* isThereFreeStoreBuffer – מחזיר true אם יש store buffer פנוי. במתודה זו נשתמש בשלב ה- issue.
* updateTags – עובר על כל הבאפרים, ואם יש באפר אשר תלוי ב- tag הרלוונטי, מעדכן את ה- value. במתודה זו נשתמש בשלב ה- write to CDB.
* isThereAddressCollision –
* freeBuffer – הופך את הבאפר הרלוונטי ל- not busy. במתודה זו נשתמש בשלב ה- write to CDB.

LoadStoreBuffer

Class זה מייצג באפר באופן כללי. יורש את ReservationStation ובכך מקבל את תכונותיו. מכיל את הכתובת המחושבת שאליה נעשה store או שממנה נעשה load, וכן שני משתנים המצייגים את value1 – הערך שיש ב- intRegisters[SRC0] ו- value2 – עבור store בלבד – הערך שנרצה לשמור בזיכרון הנמצא ב- floatRegisters[SRC1]. פונקציות תומכות:

* calculateAddress – מחשב את הכתובת הרלוונטים על פי הערך שניתן מהרגיסטר וה- imm של הפקודה.
* setValue1 – מעדכן את value1 וכן את ה- tag שלו להיות empty.
* setValue2 – מעדכן את value2 וכן את ה- tag שלו להיות empty.

LoadBuffer & StoreBuffer

Classes אלו מייצגים load & store buffers. הם יורשים מ- LoadStoreBuffer.

Package: reservationStations

ReservationStations

Class זה מכיל את כל ה- reservation stations שיש למעבד. באתחול מקבל את מספר ה- reservation stations מכל סוג על פי הקונפיגורציה של המעבד, ומאתחל בהתאם את ה- reservation stations. פונקציות תומכות:

* isThereFreeAluRS – מחזיר true אם יש ALU reservation station שאינו busy. במתודה זו נשתמש בשלב ה- issue.
* isThereFreeMulRS – מחזיר true אם יש MUL reservation station שאינו busy. במתודה זו נשתמש בשלב ה- issue.
* isThereFreeAddRS – מחזיר true אם יש ADD reservation station שאינו busy. במתודה זו נשתמש בשלב ה- issue.
* getFreeAluReservationStation – מחזיר ALU reservation station שאינו busy. במתודה זו נשתמש בשלב ה- issue.
* getFreeMulReservationStation – מחזיר MUL reservation station שאינו busy. במתודה זו נשתמש בשלב ה- issue.
* getFreeAddReservationStation – מחזיר ADD reservation station שאינו busy. במתודה זו נשתמש בשלב ה- issue.
* updateTags – עובר על כל ה- reservation stations, ואם יש reservation station אשר תלוי ב- tag הרלוונטי, מעדכן את ה- value. במתודה זו נשתמש בשלב ה- write to CDB.
* freeReservationStation – הופך את ה- reservation station הרלוונטי ל- not busy. במתודה זו נשתמש בשלב ה- write to CDB.

ReservationStation

Class זה מייצג reservation station. מכיל את ה- opcode, ואינדיקציה בנוגע להאם התחנה במצב busy. פונקציות תומכות:

* isReady – מחזיר true אם ה- instruction הרלוונטי לאותה תחנה מוכן לביצוע execute.
* Free – משחרר את ה- reservation station.

AluReservationStation & MulOrAddReservationStation

אלו הם בעצם ה- reservation stations עצמם, בסוגם השונים. Classes אלו יורשים מ- ReservationStation.

Package: units

Unit

Class זה מייצג יחידה במעבד. מכיל את ה- delay של היחידה ואת מספר ה- instructions המחכים ליחידה בתור, על מנת לדעת מתי instruction מסויים יכול להיכנס ליחידה.

integerALU & LoadStore & FPAddSub & FPMul

אלו הן היחידות עצמם, כולן יורשות מ- Unit. מכילות פונקציית execute: כאשר instruction מסויים נכנס ליחידה על מנת לבצע execute, נשתמש בפונקציה זו.

Package: registers

Registers

Class זה מכיל את כל הרגיסטרים שבתוכנית. מכיל פונקציות לעדכון הרגיסטרים / שליפה מהרגיסטרים, ולעדכון ה- tags כאשר כותבים ל- CDB.

Register

מייצג רגיסטר בודד. מכיל את הסטאטוס של הרגיסטר: האם הוא תלוי ב- tag מסויים או שיש בו ערך מעודכן.

IntRegister & FloatRegister

Classes אלו יורשים מ- Register, ומייצגים רגיסטרים מסוג int או float.

Package: exceptions

ב- package זה מימשנו exceptions שונים הנקראים מתוך התוכנית במקרה שצריך לזרוק חריג:

* AddressOutOfMemoryException
* MissingNumberOfLoadStoreBuffersException
* MissingNumberOfReservationStationsException
* ProgramCounterOutOfBoundException
* UnknownOpcodeException

Flow

כל ה- flow של התוכנית מתבצע במחלקה Tomasulo, ומשתמש במבנה הנתונים שבנינו.

השדות שבמחלקה Tomasulo:

* instructionsQueue – מכיל את כל הפקודות הנכנסות לתור בשלב ה- issue.
* instructionsStaticQueue – מכיל את כל הפקודות הנכנסות לתור בשלב ה- issue, אבל הוא לא מתרוקן, שכן הוא נועד להדפסת ה- trace בסיום התוכנית.
* waitingList – מכיל את כל הפקודות שביצעו issue, אבל עדיין אינן מוכנות לעבור לשלב ה- execute.
* executeList – מכיל את כל הפקודות שכעת בשלב ה- execute.
* writeToCDBList – מכיל את כל הפקודות שסיימו execute ומוכנות לכתוב ל- CDB (פקודות store לא ייכנסו לרשימה זו).
* Memory – הזיכרון.
* Registers – הרגיסטרים של המעבד.
* reservationStations – כל ה- reservation stations של המעבד.
* Buffers – כל הבאפרים של המעבד.
* fetchingStatus – האם עלינו לבצע fetch.
* globalStatus – האם התוכנית הסתיימה.
* fetchedHaltInst – האם ביצענו fetch לפקודת HALT.
* Clock
* Pc
* Alu\_unit – יחידת ה- ALU של המעבד.
* FP\_add\_sub\_unit – יחידת החיבור / חיסור floats של המעבד.
* FP\_mult\_unit – יחידת הכפל floats של המעבד.
* Load\_store\_unit – יחידת ה- load / store של המעבד.

אתחול:

באתחול ה- Tomasulo אנחנו נאתחל את כל מה שהוזכר לעיל בהתאם לקונפיגורציה ולזיכרון.

מהלך התוכנית:

כל עוד התוכנית לא הסתיימה, נקרא לפונקציית step (מה- main). בכל קריאה לפונקציה זו ה- clock יעלה ב- 1.

Step():

1. ביצוע fetch לפקודה הבאה.
2. אם יש פקודות ב- waitingList – ז"א יש פקודות שביצעו issue אך עדיין אינן מוכנות ל- execute, נטפל ברשימה זו: פקודות שמוכנות לבצע execute נעביר ל- executeList.
3. נשלוף את הפקודה הבאה מתוך הזיכרון: אם היא לא null