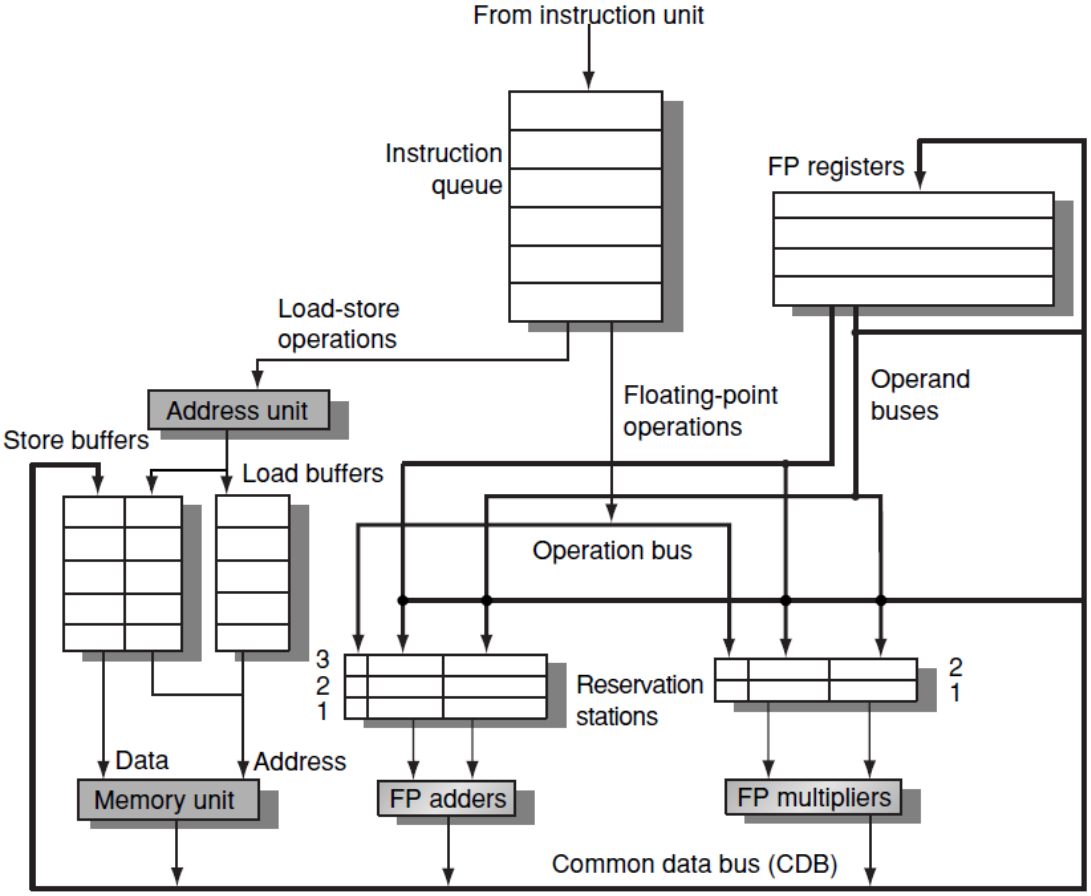
פרויקט טומסולו

בפרויקט זה נממש סימולטור של מעבד FLOATING POINT המשתמש באלגוריתם טומסולו. הפרויקט יתבצע בזוגות, להגשה עד סוף הסמסטר.



המעבד כולל:

- 16 floating point registers, שנסמנם F0-F15, כל אחד ברוחב 32 סיביות ושומר מספר נקודה צפה בפורמט single precision (ביט אחד עבור הסימן, 8 עבור האקספוננט ו־ 23 עבור המנטיסה). בתחילת הריצה כל רגיסטר מכיל מספר השווה לאינדקס שלו: F0 מכיל 0, F1 מכיל 1.0, וכך הלאה.
- מספר ניתן לקינפוג של reservation stations.
- מספר יחידות פונקציונאליות לנקודה צפה עבור חיבור/חיסור, כפל, וחילוק. היחידות לא מצונרות, ובעלות השהייה ניתנת לקנפוג. באם יש יותר הוראות מוכנות להרצה מאשר יחידות פונקציונאליות פנויות, הארביטרצייה אילו הוראות להכניס להרצה היא לבחירתכם.
- יחידת Fetch שקוראת עד שתי הוראות במקביל מזיכרון ההוראות בכל מחזור שעון אל תוך ה־ Instruction Queue, שהינו בגודל 16 הוראות, כל עוד יש מקום. במחזור שעון מספר 0 מתבצע Fetch של שתי ההוראות הראשונות וכתיבה אל תור ההוראות.
- המעבד מבצע Issue של עד שתי הוראות במקביל בכל מחזור שעון מתור ההוראות אל תוך התחנות, בהתאם לזמינות התחנות. הפענוח מתבצע בו־זמנית עם ה־ Issue, ולא לוקח מחזור שעון נוסף. למשל במחזור שעון 1 כבר מבצעים issue של שתי ההוראות הראשונות שנקראו במחזור 0 (בהנחה שיש שתי תחנות פנויות, אחרת הוראה בודדת), וכמובן שבמקביל מבצעים כבר Fetch של ההוראות הבאות.
- במעבד יש 3 CDB: אחד למחברים/מחסרים, אחד למכפלים, ואחד למחלקים. כל CDB יכול להעביר דאטא+תג בודד בכל מחזור שעון. כתיבה ל־ CDB מתבצעת מחזור שעון אחד לאחר סיום פעולת

היחידה הפונקציונאלית, במידה וה־ CDB פנוי. במידה ומספר יחידות רוצות לגשת לאותו CDB בו זמנית, מתבצעת ארביטרצייה לבחירתכם, כאשר שאר היחידות שלא זכו בארביטרצייה מחכות.

כל הוראה מקודדת ב־ 32 סיביות, בפורמט אחיד:

bits	31-28	27–24	23–20	19–16	15–12	11–0
	0	OPCODE	DST	SRC0	SRC1	0

ה־ OPCODE מתאר את ההוראה שאותה יש לבצע. שדה ה־ DST הוא רגיסטר היעד, והשדות SRC0, SRC1 הם שני רגיסטרי המקור.

כאשר סט ההוראות מכיל:

opcode name	number	description
ADD	2	F[DST] = F[SRC0] + F[SRC1]
SUB	3	F[DST] = F[SRC0] - F[SRC1]
MULT	4	F[DST] = F[SRC0] * F[SRC1]
DIV	5	F[DST] = F[SRC0] / F[SRC1]
HALT	6	exit simulator

1 סביבות תכנות:

הפרויקט ימומש בשפת C בסביבת Visual Studio במערכת ההפעלה Windows. יש להגיש את כל ספריית ה־ Solution כך שנוכל לקמפל ע"י build solution.

2 הרצה וקבצים:

הפרויקט יבנה אל תוך command line application שנקרא sim.exe, ויורץ עם רשימת קבצי טקסט בתור פרמטרים:

sim.exe cfg.txt memin.txt regout.txt traceinst.txt tracecdb.txt

כאשר cfg.txt ו־ memin.txt הינם קבצי קלט, ושאר הקבצים הינם קבצי פלט.

קובץ הקונפיגורצייה cfg.txt מכיל שורות מהצורה parameter = value, כאשר הפרמטרים הינם:

- add_nr_units = x: מספר יחידות החיבור/חיסור.
- mul_nr_units = x: מספר יחידות הכפל.
- div_nr_units = x: מספר יחידות החילוק.
- add_nr_reservation = x: מספר ה־ reservation stations עבור יחידות החיבור/חיסור.
- mul_nr_reservation = x: מספר ה־ reservation stations עבור יחידות הכפל.
- div_nr_reservation = x: מספר ה־ reservation stations עבור יחידות החילוק.
- add_delay = x: השהיית יחידות החיבור/חיסור במחזורי שעון.
- mul_delay = x: השהיית יחידות הכפל במחזורי שעון.
- div_delay = x: השהיית יחידות החילוק במחזורי שעון.

קובץ תמונת הזיכרון memin.txt מכיל 4096 שורות של תמונת הזיכרון הראשי כאשר כל שורה מכילה 32 סיביות ב־ 8 ספרות הקסאדצימליות. התוכנית מתחילה לרוץ מ־ PC=0, כאשר ההוראה שם מקודדת בשורה הראשונה בקובץ.

אם החל מכתובת מסויימת ועד הסוף תוכן הזיכרון מכיל רק אפסים, מותר לא לרשום שורות אלו, בהבנה שהזיכרון יכיל אפסים. זה נעשה רק כדי לחסוך בגודל הקבצים ואינו חובה, כלומר אפשר גם בכל מקרה לכתוב את כל 4096 השורות.

הקובץ regout.txt מכיל את פלט רגיסטרי ה־ floating point בסיום ריצת התוכנית. יהיו שם 16 שורות, כאשר כל שורה i הינה מספר עשרוני עבור תוכן הרגיסטר $F[i]$.

הקובץ traceinst.txt מכיל שורות בפורמט הבא:

instruction pc tag cycle_issued cycle_execute_start cycle_execute_end cycle_write_cdb

כאשר יש שורה עבור כל הוראה לפי סדר ה־ ISSUE (לא לפי סדר ה־ COMPLETION).

- שדה ה־ instruction הוא קידוד ההוראה בשמונה ספרות הקסא כפי שנקראו מהזיכרון.
- שדה ה־ pc הינו ה־ pc של ההוראה (0 עבור ההוראה הראשונה בזיכרון, 1 עבור השנייה וכך הלאה).
- שדה ה־ tag מכיל את התג של ההוראה, כלומר שם התחנה שמקבלת את ההוראה. שם התחנה מורכב משם היחידה הפונקציונאלית שמטפלת בתחנות, ומספר התחנה. לדוגמא תגים אפשריים הינם ADD3, MUL2, DIV1.
- שדה ה־ cycle_issued הוא מחזור השעון שבו ההוראה נכנסה לאחת התחנות.
- שדה ה־ cycle_execute_start הינו מחזור השעון שבו ההוראה התחילה להתבצע על יחידה פונקציונאלית.
- שדה ה־ cycle_execute_end הינו מחזור השעון האחרון שבו ההוראה עדיין מתבצעת ביחידה הפונקציונאלית.
- שדה ה־ write_cdb הינו מחזור השעון שבו התוצאה נכתבה על ה־ CDB.

הקובץ tracecdb.txt מכיל שורות בפורמט הבא:

cycle pc cdbname data tag

כאשר מופיעה שורה עבור כל CDB וכל מחזור שעון שבו יש בו שימוש (כלומר יתכנו עד 4 שורות בכל מחזור שעון עבור ארבעת ה־ CDB):

- שדה ה־ cycle הוא מחזור השעון שבו עובר הדאטא.
- שדה ה־ pc הינו ה־ pc של ההוראה שהסתיימה ומעבירה דאטא על ה־ CDB.
- שדה ה־ cdbname מציין לאיזה CDB השורה מתייחסת, והינו אחד משלושת האפשרויות הבאות: ADD, MUL, DIV.
- שדה ה־ data מכיל את הערך המספרי שעובר על ה־ CDB באותו מחזור שעון, בפורמט עשרוני.
- שדה ה־ tag מכיל את התג המתאים ל־ data, באותו פורמט כמו ב־ traceinst.txt.

3 דוקומנטצייה:

הקפידו שהקוד יהיה קריא, ומכיל comments לגבי מבני הנתונים והפונקציות. כמו כן יש להגיש דוקומנטצייה חיצונית המתארת באופן כללי את הפרויקט.

4 בדיקות:

הפרויקט שלכם יבדק בן השאר ע"י תוכניות בדיקה שלא תקבלו מראש. לכן חשוב מאוד לבדוק נכונות ע"י בנייה של קטעי קוד שונים, וכמו כן בדיקה עם פרמטרים שונים בקובץ הקונפיגורצייה.

יש להגיש 3 ספריות בדיקה, כאשר בכל ספרייה יהיו קבצי הקלט והפלט עבור הבדיקה.

יש לתאר בדוקומנטצייה את הבדיקות שבוצעו.