פרויקט טומסולו ספקולטיבי

בפרויקט זה נממש סימולטור של מעבד 32 ביטים המשתמש באלגוריתם טומסולו הספקולטיבי עם ROB. הפרויקט יתבצע בזוגות, להגשה עד סוף הסמסטר.

המעבד כולל:

- יחידת Fetch שקוראת הוראה חדשה מזיכרון ההוראות בכל מחזור שעון אל תוך ה־ Instruction Queue, שהינו בגודל 16 הוראות.
- לצורך חיזוי הסתעפויות, יחידת ה־ Fetch כוללת Branch Target Buffer עם 16 איברים. ה־ BTB לא כולל הסטוריית קפיצות וכל השורות בו מייצגות רק הסתעפויות שקרו.
- יחידת חיבור/חיסור מצוררת עבור FP ADD ו־ FP SUB, השהייה ניתנת לקנפוג.
- יחידת כפל מצוררת עבור FP MUL, השהייה ניתנת לקנפוג.
- יחידת עיבוד הוראות Integer ALU מצוררת, השהייה ניתנת לקנפוג. חישוב כתובות לזיכרון, והשוואות לצורך קפיצות, מתבצע ביחידת ה־ Integer.
- Reorder Buffer עם עומק ניתן לקנפוג.
- 16 general purpose integer registers, שנסמנם R0-R15, כל אחד ברוחב 32 סיביות. בתחילת הריצה כל הרגיסטרים מאותחלים לאפס.
- 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 לקנפוג של reservation stations, load buffers, store buffers.
- הניחו שיש מספר CDB כך שאין התנגשות בין היחידות הפונקציונאליות. הכתיבה ל־ CDB מתבצעת מחזור שעון אחד אחרי סיום ה־ Execution ביחידה הפונקציונאלית.
- הזיכרון הראשי בגודל 1024 מילים של 32 סיביות כל אחד.

כל הוראה מקודדת ב־ 32 סיביות, בפורמט אחיד:

bits	31–28	27–24	23–20	19–16	15–0
	OPCODE	DST	SRC0	SRC1	IMM (signed)

ה־ OPCODE מתאר את ההוראה שאותה יש לבצע. שדה ה־ DST הוא רגיסטר היעד, והשדות SRC0, SRC1 הם שני רגיסטרי המקור. השדה IMM הוא קבוע בן 16 סיביות במשלים ל־ 2 (יכול לייצג גם מספר שלילי).

כאשר סט ההוראות מכיל:

opcode name	number	description
LD	0	F[DST] = MEM[R[SRC0] + IMM]
ST	1	MEM[R[SRC0] + IMM] = F[SRC1]
JUMP	2	unconditional branch to PC + IMM
BEQ	3	if R[SRC0] == R[SRC1] branch to PC + IMM
BNE	4	if R[SRC0] != R[SRC1] branch to PC + IMM
ADD	5	R[DST] = R[SRC0] + R[SRC1]
ADDI	6	R[DST] = R[SRC0] + IMM
SUB	7	R[DST] = R[SRC0] - R[SRC1]
SUBI	8	R[DST] = R[SRC0] - IMM
ADD.S	9	F[DST] = F[SRC0] + F[SRC1]
SUB.S	10	F[DST] = F[SRC0] - F[SRC1]
MULT.S	11	F[DST] = F[SRC0] * F[SRC1]
HALT	12	exit simulator

בכל הגישות לזיכרון, הכתובת לזיכרון היא של מילים 32 סיביות (לא של בתים).

בצוע Fetch של הוראה מהזיכרון לוקח מחזור שעון אחד, שבמהלכו ההוראה נכתבת ל־ Instruction Queue (עומק 16), ומעדכנים את ה־ PC בהתאם לחזאי הקפיצות. יחידת ה־ Fetch ממשיכה להביא הוראות חדשות באופן ספקולטיבי גם לפני שידועה תוצאת הקפיצה. במידה ומתברר בהמשך שהחזאי טעה, מעדכנים את ה־ BTB, מרוקנים את ה־ Instruction Queue, ומביאים הוראות החל מהכתובת הנכונה.

כאשר ההוראה נקראת מתור ההוראות, מתבצע decoding של ההוראה ו־ Issue אל תוך reservation station במידה ויש תחנה פנויה.

1 סביבות תכנות:

ניתן לממש את הפרויקט בשפת C, או ++C, או Java. ניתן לממש על סביבת Windows או Linux. אם משמשים ב־ Visual Studio, יש להגיש את כל ספריית ה־ Solution כך שנוכל לקמפל ע"י build solution. ב־ Linux יש לכלול Makefile שבונה את הפרויקט כאשר מריצים make.

2 הרצה וקבצים:

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

sim cfg.txt memin.txt memout.txt regint.txt regout.txt trace.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:

- int_delay = x: השהיית יחידת ה־ Integer ALU במחזורי שעון.
- add_delay = x: השהיית יחידת החיבור/חיסור עבור הוראות נקודה צפה במחזורי שעון.
- mul_delay = x: השהיית יחידת הכפל עבור הוראות נקודה צפה במחזורי שעון.
- mem_delay = x: השהיית קריאה/כתיבה לזיכרון במחזורי שעון.
- rob_entries = x: גודל ה־ ROB.
- add_nr_reservation = x: מספר ה־ reservation stations עבור יחידת החיבור/חיסור.
- mul_nr_reservation = x: מספר ה־ reservation stations עבור יחידת הכפל.
- int_nr_reservation = x: מספר ה־ reservation stations עבור יחידת ה־ Integer ALU.
- mem_nr_load_buffers = x: מספר ה־ load buffers.
- mem_nr_store_buffers = x: מספר ה־ store buffers.

קובץ תמונת הזיכרון memin.txt מכיל 1024 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.

הקובץ memout.txt הינו באותו הפורמט כמו memin.txt, ומכיל את תמונת הזיכרון בסיום הרצת התוכנית.

הקובץ regint.txt מכיל את פלט רגיסטרי ה־ integer בסיום ריצת התוכנית. יהיו שם 16 שורות, כאשר כל שורה i הינה תוכן הרגיסטר $R[i]$ בפורמט decimal.

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 trace.txt מכיל שורות בפורמט הבא:

instruction cycle_issued cycle_execute_start cycle_write_cdb cycle_commit

כאשר יש שורה עבור כל הוראה שבוצע עבורה issue (כולל הוראות ספקולטיביות) לפי סדר פענוח ההוראות.

- שדה ה־ instruction הוא קידוד ההוראה בשמונה ספרות הקסא כפי שנקראו מהזיכרון.
- 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־ write_cdb הינו מחזור השעון שבו התוצאה נכתבה על ה־ CDB, או -1 אם זה לא רלוואנטי להוראה.
- שדה ה־ cycle_commit הינו מחזור השעון שבו בוצע commit להוראה.

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.