ת"ב 4 – סימולטור Multithreading

בתרגיל בית זה תממשו סימולטור המדמה ריצה של מעבד מרובה חוטים בשתי תצורות:Block Multithreading (MT) ו-Fine-grained MT.

מיקרוארכיטקטורה

לשם פשטות, שתי המיקרו-ארכיטקטורות בבסיסן הן Single-Cycle, כך שפקודה לוקחת מחזור שעון יחיד בהינתן אלא אם מדובר בפקודת Load או Store. הארכיטקטורה תומכת בסט הפקודות הבאה:

- פקודות גישה לזיכרון: LOAD, STORE
- פקודות אריתמטיות: ADD, ADDI, SUB, SUBI
- HALT - פקודה מיוחדת אשר תשמש לעצירת החוט שרץ כרגע (גם כן לוקחת מחזור יחיד).

בנוסף נתון:

- בארכיטקטורה הנתונה 8 רגיסטרים כלליים, R0 – R7.
- פקודות אריתמטיות יבוצעו בין רגיסטרים או בין רגיסטר למספר קבוע.
- פקודות גישה לזכרון יקחו מספר מחזורי שעון (מוגדר בקובץ הפרמרטרים).
- בכל מחזור שעון מתבצעת לכל היותר פקודה אחת ($IPC_{max} = 1$).
- מרחב הכתובות של כל החוטים זהה. <u>ניתן להניח כי אין תלויות מידע בין חוטים.</u>
- מרחב הכתובות הוירטואליות זהה למרחב הכתובות הפיזיות (איננו מתייחסים ל-Virtual Memory בתרגיל זה).
- עבור תצורה של Fine-grained MT:
  - החלפת ההקשר נעשת כל מחזור שעון אחד.
  - אין קנס (penalty) בהחלפת הקשר.
- עבור תצורה של Blocked MT
  - קיים קנס בהחלפת הקשר (מוגדר בקובץ הפרמרטרים).

הממשק לסימולטור

עליכם לממש את הפונקציות המוגדרות בקובץ core_api.h. על הסימולטור לעקוב אחר הלוגיקה של התוכנית ועל ביצועי המכונה, כלומר על ערכי הרגיסטרים וזמן ריצת התוכנית במחזורי שעון. לשם כך, עליכם לממש את הפונקציות הבאות:

- `CORE_BlockedMT()` – הפונקציה תכיל סימולציה מלאה של מכונת Blocked MT. למעשה, הפונקציה תחזור כאשר כל החוטים יגיעו לפקודה HALT. במעמד זה מבני הנתונים שתיצרו יכילו את ה-register files ואת כמות מחזורי השעון אשר לקח לתוכנית לרוץ.
- `CORE_BlockedMT_CTX(tcontext context[], int threaded)` – הפונקציה תחזיר דרך מצביע context עבור thread ספציפי את מצב ה-register file.
- `CORE_BlockedMT_CPI()` – הפונקציה מחזירה את ביצועי המערכת במדד CPI. ניתן להניח כי זו הפונקציה האחרונה שנקרא לה בסימולציה (עבור Blocked MT) כך שניתן לשחרר בנקודה זו את הקצאות הזכרון.

הפונקציות הנ"ל קיימות גם עבור תצורת Fine-grained MT.

המימוש שלכם ייכתב בקובץ בשם core_api.c או core_api.cpp, לאלו אשר מעדיפים לממש ב-C++. שימו-לב שגם עבור מימוש ב-C++ עליכם לחשוף ממשק C, כפי שמוגדר בקובץ core_api.h.

גישה לזיכרון

מערכת הזיכרון נתונה. הממשק לסימולטור מוגדר בקובץ sim_api.h והמימוש בקובץ sim_api.c. הממשק לזיכרון מאפשר לסימולטור שלכם לקרוא פקודות ולקרוא/לכתוב נתונים. מכיוון שאנו מניחים כי אין תלויות מידע בין חוטים, זמן הכתיבה ו/או הקריאה לזיכרון בפועל איננו רלוונטי. לדוגמה, אם נתון בקובץ ההגדרות כי כתיבה לזיכרון לוקחת 5 מחזורי שעון, לא משנה האם בפועל המידע נכתב במחזור שעון 0, 3 או 5, לדוגמה, שכן אותו חוט יהיה גם כך במצב idle, ושאר החוטים לא יקראו מאותה כתובת (הנחת התרגיל).

את תוכן הזיכרון ניתן לאתחל מקובץ מפת הזיכרון. קבצי דוגמה למפת הזיכרון לטעינה כלולים בחומרי התרגיל (קבצים עם סיומת img) וכוללים גם תיעוד מבנה הקובץ בהערות. קובץ מפת הזיכרון מכיל הן פקודות לביצוע והן נתונים לקריאה/כתיבה. סימולטור הזיכרון מוגבל להכיל 100 פקודות עבור כל חוט ו-100 נתונים עוקבים.

חוטים

כל מעבד יריץ מספר חוטים כפי שנקבע בקובץ ה-img. החלפת חוטים מתבצעת בשיטת Round-Robin (RR) עם איתחול לחוט 0. אם חוט כלשהו סיים (הגיע לפקודה HALT), מדלגים עליו. לדוגמה, אם חוט 1 סיים ונשארו במערכת חוטים 0, 2 ו-3, אז אחרי חוט 0 יגיע תורו של חוט 2 (במידה והוא כמובן יכול לרוץ).

בהקשר של Fine-grained MT, המימוש הינו בתצורה Flexible. המשמעות היא כי מנגנון ה-Round-Robin לא בוחר חוט אשר הגיע ל-HALT ומבזבז עליו מחזור שעון.

חוט שמגיע לפקודת halt הוא חוט שסיים את עבודתו. **פקודת halt לוקחת מחזור שעון.**

דוגמה

לבהירות התרגיל, להלן טבלת מעקב עבור קובץ קלט example1.img בהינתן תצורות Blocked MT ו-Fine-grained MT:

Example 1: Blocked MT Simulation			
Cycle	Thread	Command	Description
0	0	HALT	
1			Switch Overhead (0->1)
2	1	LOAD	
3			Switch Overhead (1->2)
4	2	STORE	
5			Idle
6			Idle
7			Idle
8	2	LOAD	
9			Switch Overhead (2->1)
10	1	ADD	
11	1	ADD	
12	1	HALT	
13			Idle
14			Switch Overhead (1->2)
15	2	ADDI	
16	2	ADD	
17	2	HALT	

Cycles = 18 **Instructions = 10**
CPI = 1.8

Example 1: Fine-grained MT Simulation			
Cycle	Thread	Command	Description
0	0	HALT	
1	1	LOAD	
2	2	STORE	
3			Idle
4			Idle
5			Idle
6	2	LOAD	
7	1	ADD	
8	1	ADD	
9	1	HALT	
10			Idle
11			Idle
12	2	ADDI	
13	2	ADD	
14	2	HALT	

Cycles = 15 **Instructions = 10**
CPI = 1.5

נקודות לשים לב אליהן:

1. ה-RR מאותחל תמיד לחוט 0 ובוחר בצורה ציקלית את החוט הבא.
2. פקודות HALT לוקחות מחזור שעון.
3. פקודות LOAD ו-STORE לוקחות מחזור שעון ורק לאחר מכן מתחילה ספירת ה-latency מול הזיכרון (בהתאם לנתון בקובץ ההגדרות) – מומחש היטב בטבלאות המעקב לעיל.
4. ב-Blocked MT, במחזור שעון 8, גם חוט 1 וגם חוט 2 מוכנים לפעולה, אך ה-RR בוחר בחוט 2 שכן זה החוט האחרון עליו המכונה עבדה.
5. ב-Blocked MT, במחזור שעון 13, המכונה במצב idle שכן פקודת ה-LOAD ממחזור שעון 8 (חוט 2) עדיין עובדת מול הזיכרון. במחזור שעון 14 מתבצעת החלפת הקשר, שכן רק במחזור שעון 14 חוט 2 מוכן בפועל.
6. ב-Fine-grained MT, ברגע שחוט 0 מסתיים, ה-RR מדלג עליו ולא מבזבז עליו מחזורי שעון (Flexible).

סביבת בדיקה

קובץ ה-makefile נתון, לנוחיותכם. לאחר הבניה יתקבל קובץ הרצה בשם `sim_main`.

את הקובץ המצורף ניתן להפעיל באופן הבא:

```
./sim_main <test_file>
```

לדוגמא:

```
./sim_main example1.img
```

שימו-לב: ה- `main` שניתן נועד להקל עליכם בבדיקה, אולם אתם מחויבים למימוש **הממשק** לסימולטור כפי שמוגדר ב-core_api.h . כלומר, ייתכן והסימולטור ייבדק בדרכים **שונות** מהמודגם ה-`main` וייתכן שימוש בקובץ `main` אחר מהמסופק, אשר משתמש באותו הממשק. לכן הקפידו שהמימוש שלכם יעמוד בדרישות המוגדרות בתרגיל.

דרישות ההגשה

הגשה אלקטרונית בלבד באתר הקורס ("מודל") מחשבונו של אחד הסטודנטים.

מועד ההגשה: עד ה-30.6.2022 בשעה 23:55.

אין לערוך שינוי באף אחד מקבצי העזר המסופקים לכם. ההגשה שלכם לא תכלול את אותם קבצים, מלבד המימוש שלכם ב-`core_api.c/cpp` ותיבדק עם גרסה של סביבת הבדיקה של הבודק. עמידה בדרישות הממשק כפי שמתועדות בקובץ הממשק (core_api.h) היא המחייבת.

עליכם להגיש קובץ *tar.gz בשם hw4_*ID1_ID2*.tar.gz כאשר ID1 ו-ID2 הם מספרי ת.ז. של המגישים. לדוגמה: hw4_012345678_987654321. tar.gz. ה-tar יכיל קובץ בודד:

- קוד המקור של הסימולטור שלכם: `core_api.c` **או** `core_api.cpp`.
  קוד המקור חייב להכיל תיעוד פנימי במידה סבירה על מנת להבינו.

הוראות ליצירת קובץ ה-tar.gz ניתן למצוא בהוראות בתרגילי הבית הקודמים.

דגשים להגשה:

1. המימוש שלכם **חייב** להתקמפל בהצלחה ולרוץ במכונה הוירטואלית שמסופקת לכם באתר המודל של הקורס. זוהי סביבת הבדיקה המחייבת לתרגילי הבית. **כל בעיה בהגשה המונעת את הרצת הקוד (כיווץ לא נכון של הקבצים, קוד לא מתקמפל, ...) יגרור ציון 0!**
2. מניסיונם של סטודנטים אחרים: הקפידו לוודא שהקובץ שהעלתם לאתר הקורס הוא אכן הגרסה שהתכוונתם להגיש. לא יתקבלו הגשות נוספות לאחר מועד ההגשה בטענות כגון "הקובץ במודל לא עדכני ויש לנו גרסה עדכנית יותר שלא נקלטה."

העתקות יטופלו בחומרה

בהצלחה!