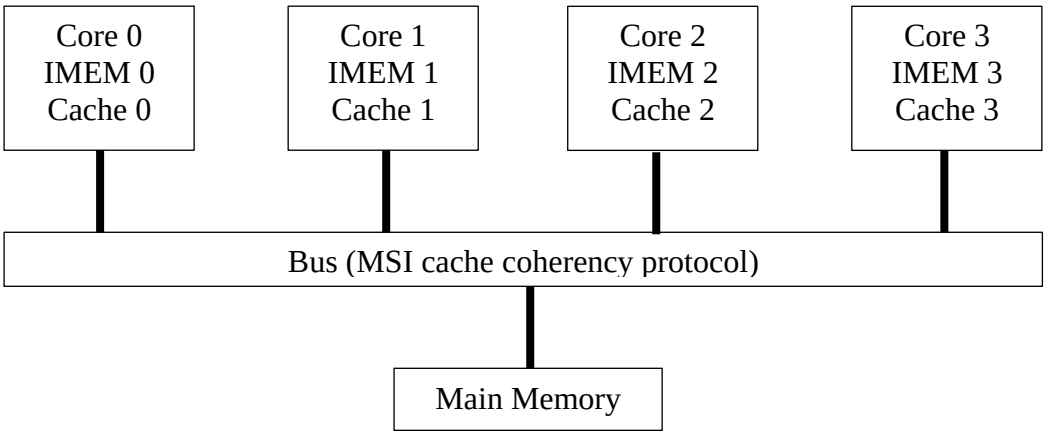
אוניברסיטת תל-אביב, הפקולטה להנדסה

פרויקט בקורס: ארכיטקטורה של מחשבים 0512.4461

שנת הלימודים תשפ"א, סמסטר א'

בפרויקט נממש סימולטור של מעבד הכולל 4 ליבות אשר רצות במקביל:



כל ליבה הינה מצוררת, כוללת זיכרון הוראות SRAM פרטי, ומטמון נתונים פרטי. הליבות מחוברות באמצעות BUS העובד בפרוטוקול קוהרנטיות MSI לזיכרון ראשי משותף.

רגיסטרים

כל ליבה מכילה 16 רגיסטרים, שכל אחד מהם ברוחב 32 ביטים.

רגיסטר מספר 1 הינו רגיסטר מיוחד שלא ניתן לכתוב אליו, ותמיד מכיל את שדה ה- immediate, לאחר בצוע sign extension, כפי שקודד בהוראת האסמבלי. הוא מתעדכן עבור כל הוראה כחלק מפענוח ההוראה. רגיסטר 0 הינו זהותית אפס. הוראות אשר כותבות לרגיסטר 0 לא משנות את ערכו.

זיכרון הוראות

לכל ליבה זיכרון הוראות SRAM פרטי ברוחב 32 סיביות ובעומק 1024 שורות. רגיסטר ה- PC לכן הינו ברוחב 10 ביטים, והוראות עוקבות מקדמות את PC באחד.

צנרת

בכל ליבה יש שימוש ב- delay slot, והצנרת כוללת 5 שלבים:

Fetch	Decode	Execute	Mem	Write Back

אין שימוש במעקפים.

בניגוד לצנרת מיפס שלמדנו, אין שימוש בחצאי מחזורי שעון בגישה למערך הרגיסטרים אלא במחזור שעון שלם. בכל מחזור שעון ניתן לבצע שלוש קריאות ממערך הרגיסטרים + כתיבה במקביל, אבל את הדאטא שכותבים במחזור שעון מסוים נקבל בקריאה רק במחזור השעון הבא.

Branch Resolution מתבצע בשלב הפענוח.

ההוראות נקראות מזיכרון הוראות SRAM פנימי לליבה במחזור שעון בודד.

פיתרון Data Hazards בין הוראות מתבצע ע״י Stall של ההוראה התלויה בשלב הפענוח.

במקרה של החטאה במטמון הנתונים, מתבצע Stall של ההוראה שפנתה למטמון בשלב ה- Mem.

מטמון נתונים

בכל ליבה יש מטמון נתונים במיפוי ישיר בגודל 256 מילים. זמן הפגיעה במטמון הינו מחזור שעון בודד, גודל הבלוק הינו מילה בודדת, ומדיניות הכתיבה הינה write back, write allocate. המטמון ממומש באמצעות שני זכרונות SRAM : הזיכרון הראשון, DSRAM, הינו ברוחב 32 ביטים ובעומק 256 מילים, ומכיל את הדאטא השמור במטמון.

הזיכרון השני, TSRAM, מכיל 256 שורות, כאשר כל שורה מכילה עבור כל בלוק במטמון את התג ואת מצב פרוטוקול הקוהרנטיות MSI : Invalid – 0, Shared – 1, Modified – 2.

13:12	11:0
MSI	Tag

בתחילת הריצה זכרונות ה- DSRAM ו- TSRAM מאופסים.

זכרון ראשי ו- MSI BUS

הגישה לנתונים הינה למילים בלבד (אין תמיכה בבתים). מרחב כתובות הנתונים הינו 20 ביטים, וגודל הזיכרון הראשי הינו 2 בחזקת 20 מילים.

ה- BUS בין הליבות לזיכרון הראשי מכיל את הקווים הבאים :

bus_origid	3 bits	Originator of this transaction 0: core 0 1: core 1 2: core 2 3: core 3

		4: main memory
bus_cmd	2 bits	0: no command 1: BusRd 2: BusRdX 3: Flush
bus_addr	20 bits	word address
bus_data	32 bits	word data

בכל מחזור שעון על קווי הבס יש טראנסקציה בודדת. במידה ומספר מטמונים רוצים לפנות לבס, מתבצעת ארביטרציה כאשר לליבה באינדקס נמוך יותר יש עדיפות גבוהה יותר. לא תינתן גישה לבס לליבה מסויימת כל עוד טראנסקציה קודמת מסוג BusRd או BusRdX לא הסתיימה ע״י טראנסקציית Flush.

מי שיוצר את הטראנסקצייה וכותב לקווי הבס במחזור שעון זה (ליבה או הזיכרון הראשי) ממלא קוד בקו ה- bus_origid.

הוראות BusRd ו- BusRdX מעבירות כפרמטר את כתובת המילה בזיכרון הראשי. במידה והדאטא עדכני בזיכרון הראשי, הזיכרון הראשי יחזיר את המילה באמצעות הוראת Flush. המילה תוחזר בהשהייה של 64 מחזורי שעון לאחר קבלת הוראת הקריאה. הדאטא יוחזר בקווי ה- bus_data, והכתובת ב- bus_addr תהיה אותה הכתובת של הבקשה.

במידה והדאטא העדכני נמצא במטמון של ליבה אחרת במצב Modified, ליבה זו תחזיר את הבלוק באמצעות הוראות Flush והזיכרון הראשי יתעדכן במקביל.

סט ההוראות וקידודם

לכל ליבה יש פורמט אחיד לקידוד ההוראות לפי חלוקת הביטים הבאה:

31:24	23:20	19:16	15:12	11:0
Opcode	rd	rs	rt	immediate

האופקודים הנתמכים ע״י המעבד ומשמעות כל הוראה נתונים בטבלה הבאה:

Opcode Number	Name	Meaning
0	add	R[rd] = R[rs] + R[rt]
1	sub	R[rd] = R[rs] – R[rt]
2	and	R[rd] = R[rs] & R[rt]
3	or	R[rd] = R[rs] | R[rt]
4	xor	R[rd] = R[rs] ^ R[rt]
5	mul	R[rd] = R[rs] * R[rt]
6	sll	R[rd] = R[rs] << R[rt]
7	sra	R[rd] = R[rs] >> R[rt], arithmetic shift with sign extension
8	srl	R[rd] = R[rs] >> R[rt], logical shift
9	beq	if (R[rs] == R[rt]) pc = R[rd][low bits 9:0]
10	bne	if (R[rs] != R[rt]) pc = R[rd] [low bits 9:0]
11	blt	if (R[rs] < R[rt]) pc = R[rd] [low bits 9:0]
12	bgt	if (R[rs] > R[rt]) pc = R[rd] [low bits 9:0]
13	ble	if (R[rs] <= R[rt]) pc = R[rd] [low bits 9:0]
14	bge	if (R[rs] >= R[rt]) pc = R[rd] [low bits 9:0]
15	jal	R[15] = next instruction address, pc = R[rd][9:0]
16	lw	R[rd] = MEM[R[rs]+R[rt]]
17	sw	MEM[R[rs]+R[rt]] = R[rd]
18	ll	load linked R[rd] = MEM[R[rs]+R[rt]] set core access watch flag on the address
19	sc	store conditional check if another core wrote to the same address on watch using sc command since the ll command of this core. If there was no successful sc command from another core since the ll command, perform MEM[R[rs]+R[rt]] = R[rd] and set R[rd] = 1 (success) Otherwise, do not write to memory and set R[rd] = 0 (failure)
20	halt	Halt this core Exit simulator when all cores reached halt and the pipelines are empty

סימולטור

הסימולטור מסמלץ את צנרות הליבות, המטמונים, והזיכרון הראשי. בתחילת הריצה כל ליבה מתחילה לרוץ החל מ- PC=0. סיום הריצה ויציאה מהסימולטור מתבצע כאשר כל הליבות בצעו את הוראת ה- HALT והצנרות התרוקנו.

הסימולטור יכתב בשפת C ויקומפל לתוך command line application אשר מקבל 27 command line parameters לפי שורת ההרצה הבאה:

sim.exe imem0.txt imem1.txt imem2.txt imem3.txt memin.txt memout.txt regout0.txt regout1.txt regout2.txt regout3.txt core0trace.txt core1trace.txt core2trace.txt core3trace.txt bustrace.txt dsram0.txt dsram1.txt dsram2.txt dsram3.txt tsram0.txt tsram1.txt tsram2.txt tsram3.txt stats0.txt stats1.txt stats2.txt stats3.txt

בנוסף יש לתמוך גם בהרצת sim.exe ללא פרמטרים, ואז שמות הקבצים נלקחים כ- default לפי השמות לעיל, מאותה ספרייה שבה נמצא הקובץ sim.exe.

הקבצים **imem0.txt – imem3.txt** הינם קבצי קלט בפורמט טקסט אשר כל אחד מהם מכיל את תוכן זיכרון ההוראות של הליבה המתאימה בתחילת הריצה. כל שורה בקובץ מכילה תוכן שורה בזיכרון ההוראות, החל מכתובת אפס, בפורמט של 8 ספרות הקסאדצימליות. במידה ומספר השורות בקובץ קטן מ- 1024, ההנחה הינה ששאר הזיכרון מעל הכתובת האחרונה שאותחלה בקובץ, מאופס. ניתן להניח שקובץ הקלט תקין.

הקובץ **memin.txt** הינו קובץ קלט בפורמט טקסט אשר מכיל את תוכן הזיכרון הראשי בתחילת הריצה. כל שורה בקובץ מכילה תוכן שורה בזיכרון, החל מכתובת אפס, בפורמט של 8 ספרות הקסאדצימליות. במידה ומספר השורות בקובץ קטן מ- 2 בחזקת 20 ההנחה הינה ששאר הזיכרון מעל הכתובת האחרונה שאותחלה בקובץ, מאופס. ניתן להניח שקובץ הקלט תקין.

הקובץ **memout.txt** הינו קובץ פלט, באותו פורמט כמו memin.txt, שמכיל את תוכן הזיכרון הראשי בסיום הריצה.

הקבצים **regout0.txt – regout3.txt** הינם קבצי פלט, שמכילים את תוכן הרגיסטרים R2-R15 של הליבה המתאימה בסיום הריצה (שימו לב שאין להדפיס את הקבועים R0 ו- R1). כל שורה תיכתב באותו פורמט כמו שורה ב- dmemin.txt, 8 ספרות הקסאדצימליות.

הקבצים **core0trace.txt – core3trace.txt** הינם קבצי פלט, המכילים מעקב אחר מצב הצנרת. עבור כל ליבה, בכל מחזור שעון שלפחות אחד מהשלבים בצנרת פעיל, תודפס שורת טסקט המציינת איזו הוראה נמצאת בכל שלב בצנרת, וכמו כן תוכן מערך הרגיסטרים בפורמט הבא:

CYCLE FETCH DECODE EXEC MEM WB R2 R3 R4 R5 R6 R7 R8 R9 R10 R11 R12 R13 R14 R15

השדה CYCLE הינו מספר מחזור השעון ומודפס בבסיס דצימלי.

השדות FETCH עד WB מכילים את ה- PC של ההוראה שנמצאת בשלב המתאים בצנרת ומודפסים ב- 3 ספרות הקסאדצימליות. במידה ושלב הצנרת אינו פעיל (למשל בעת מילוי הצנרת או כאשר יש בועית בצנרת בשלב זה) יש להדפיס שלושה סימני מינוס (---).

אח"כ יש את תוכן מערך הרגיסטרים (החל מרגיסטר R2) בתחילת מחזור השעון (ה- Q של הפליפלופים, לא כולל שינויים במחזור שעון זה), כאשר כל רגיסטר מודפס ב- 8 ספרות הקסאדצימליות.

הקובץ **bustrace.txt** הינו קובץ פלט, שמכיל את תוכן קווי ה- BUS בכל מחזור שעון שבו bus_cmd שונה מאפס בפורמט הבא:

CYCLE bus_origid bus_cmd bus_addr bus_data

השדה CYCLE הינו מספר מחזור השעון ומודפס בבסיס דצימלי. שאר השדות מודפסות בהקסאדצימלי (שדות bus_origid ו- bus_cmd בספרה בודדת, bus_addr ב- 5 ספרות ו- bus_data ב- 8 ספרות).

הקבצים **dsram0.txt – dsram3.txt** הינם קבצי פלט, באותו פורמט כמו memin.txt, שכל אחד מהם מכיל את תוכן זיכרון ה- DSRAM של המטמון בליבה המתאימה בסיום הריצה, כאשר כל שורה מודפסת ב- 8 ספרות הקסאדצימליות.

הקבצים **tsram0.txt – tsram3.txt** הינם קבצי פלט, באותו פורמט כמו memin.txt, שכל אחד מהם מכיל את תוכן זיכרון ה- TSRAM של המטמון בליבה המתאימה בסיום הריצה, כאשר כל שורה מודפסת ב- 4 ספרות הקסאדצימליות.

הקבצים **stats0.txt – stats3.txt** הינם קבצי פלט, אשר מכילים סטטיסטיקות עבור כל ליבה. כל קובץ מכיל מספר שורות, כל שורה מכילה שם ומונה X בפורמט הבא, כאשר X דצימאלי:

row contents	explanation
cycles X	number of clock cycles the core was running till halt
instructions X	number of instructions executed
read_hit X	number of cache read hits
write_hit X	number of cache write hits
read_miss X	number of cache read misses
write_miss X	number of cache write misses
decode_stall X	number of cycles a pipeline stall was inserted in decode stage
mem_stall X	number of cycles a pipeline stall was inserted in mem stage

דרישות הגשה

1. יש להגיש קובץ דוקומנטציה של הפרויקט, חיצוני לקוד, בפורמט pdf, בשם project1_id1_id2_id3.pdf כאשר id1,id2,id3 הם מספרי תעודת הזהות שלכם.
2. הפרויקט יכתב בשפת התכנות סי. יש להקפיד שיהיו הערות בתוך הקוד המסבירות את פעולתו.
3. יש להגיש את הקוד ב- visual studio בסביבת windows. יש להגיש את קובץ ה- solution, ולוודא שהקוד מתקמפל ורץ, כך שניתן יהיה לבנות אותו ע״י לחיצה על build solution. יש להגיש גם את ספריית ה- build כולל קובץ ה- executable הבנוי.
4. תוכניות בדיקה. הפרויקט שלכם יבדק בין השאר ע״י תוכניות בדיקה שלא תקבלו מראש, וגם ע״י תוכניות בדיקה שאתם תכתבו באסמבלי. יש להגיש שלוש תוכניות בדיקה:

   א. תוכנית המבצעת כפל של שתי מטריצות בגודל 16x16 (כל איבר במטריצת התוצאה הינו מכפלה סקלארית של שורה בעמודה), אשר רצה על ליבה בודדת, ליבה מספר 0. ערכי המטריצה הראשונה נמצאים בכתובות 0 עד 0xFF בזיכרון הראשי, והמטריצה השנייה בכתובות 0x100 עד 0x1FF. מטריצת התוצאה תיכתב לכתובות 0x200 עד 0x2FF. כל מטריצה מסודרת בזיכרון כמו בשפת סי: ערכי שורה ראשונה משמאל לימין, ואז עוברים לשורה הבאה וכך הלאה. ניתן להניח שערכי המטריצות קטנים מספיק כך שלא יתרחש overflow. **ינתן ניקוד גבוה יותר לזמן ריצה כולל נמוך יותר.**

   ב. בצוע כפל של אותן המטריצות כמו בתוכנית בדיקה א׳, אבל על כל 4 הליבות במקביל, כאשר כל ליבה מבצעת חלק מהחישוב. **ינתן ניקוד גבוה יותר לזמן ריצה כולל נמוך יותר.**

   ג. תוכנית אשר רצה על 4 ליבות במקביל, כאשר כל ליבה מקדמת את אותה כתובת 0 בזיכרון הראשי 128 פעמים (כך שבסיום ריצת 4 הליבות הערך יהיה 512). יש להשתמש בהוראות load linked ו- store conditional. כמו כן בסיום הריצה יש לוודא כי הערך בכתובת 0 מעודכן בזיכרון הראשי (למשל ע״י אילוץ conflict miss באמצעות קריאה של ערכים אחרים לאותו האינדקס במטמונים שיוודאו שהערך נכתב חזרה לזיכרון).

5. את תוכניות הבדיקה יש להגיש בשלוש תתי-ספריות בשמות:

   mulserial, mulparallel, counter

כל ספרייה תכיל 28 קבצים : את קובץ ההרצה sim.exe, את קבצי הקלט הדרושים להרצה, וכמו כן את קבצי הפלט שקיבלתם :

sim.exe imem0.txt imem1.txt imem2.txt imem3.txt memin.txt memout.txt regout0.txt regout1.txt regout2.txt regout3.txt core0trace.txt core1trace.txt core2trace.txt core3trace.txt bustrace.txt dsram0.txt dsram1.txt dsram2.txt dsram3.txt tsram0.txt tsram1.txt tsram2.txt tsram3.txt stats0.txt stats1.txt stats2.txt stats3.txt