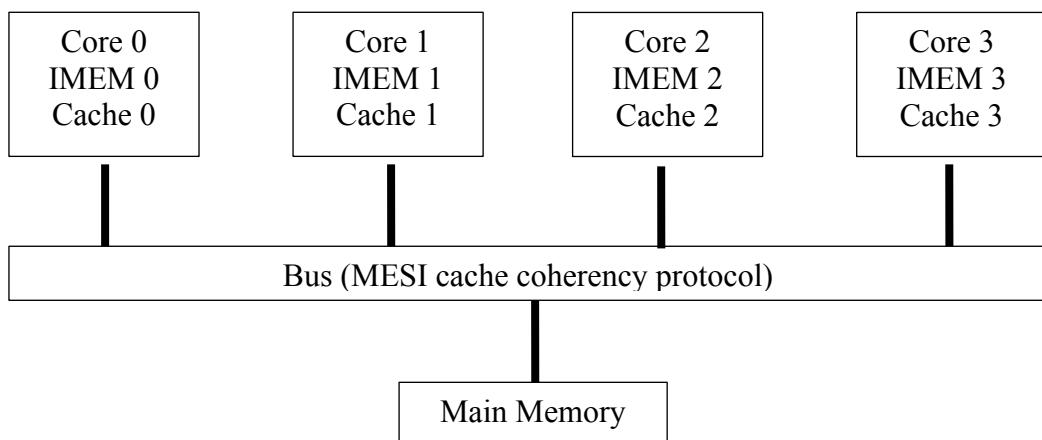


# אוניברסיטת תל-אביב, הפקולטה להנדסה

## פרויקט בקורס: ארכיטקטורה של מחשבים 0512.4461

שנת הלימודים תשפ"ו, סמסטר א'

בפרויקט נमמש סימולטור של מעבד הכלל 4 ליבות אשר רצוט במקביל:



כל ליבה הינה מצונררת, כולל זיכרון הוראות SRAM פרטני, ומטען נתונים פרטני. הליבות מחוברות באמצעות BUS העובד בפרוטוקול קוחרנטיות MESI לזכרון הראשי משותף.

### רגיסטרים

כל ליבה מכילה 16 רגיסטרים, שכל אחד מהם ברוחב 32 ביטים. הרגיסטר מס' 1 הינו רגיסטר מיוחד שלא ניתן לכתוב אליו, ותמיד מכיל את שדה ה-immediate, לאחר בוצע extension sign, כפי שקיים בהוראת האSEMBLER. הוא מתעדכן עבור כל הוראה כחלק מפענוח ההוראה. רגיסטר 0 הינו זהותית אפס. הוראות אשר כתובות לרוגיסטר 0 לא משנהות את ערכו.

### זיכרון הוראות

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

### Ճנרת

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

|       |        |         |     |            |
|-------|--------|---------|-----|------------|
| Fetch | Decode | Execute | Mem | Write Back |
|-------|--------|---------|-----|------------|

אין שימוש במקפים.

בניגוד לצנרת מיפס שלמדנו, אין שימוש בחצאי מחזרי שעון בגישה למערך הרגיסטרים אלא במחזר שעון שלם. בכל מחזר שעון ניתן לבצע שלוש קריאות למערך הרגיסטרים + כתיבה במקביל, אבל את הדטה שכותבים במחזר שעון מסוים נקבל בקריאה רק במחזר השעון הבא.

ה呼וראות נקראות מזיכרון הוראות SRAM פנימי ליבת במחזר שעון בודד. Branch Resolution

ההוראות נקראות מזיכרון הוראות SRAM פנימי ליבת במחזר שעון בודד.

פתרונות Data Hazards בין הוראות מתבצע ע"י Stall של ההוראה התלויה בשלב הפענוח.

במקרה של החטאה במתomon הנתונים, מתבצע Stall של ההוראה שפנתה למטרון בשלב ה-Mem.

## מטרון נתונים

בכל ליבת יש מטרון נתונים במיפוי ישיר בגודל 512 מילימ. זמן הפגיעה במתomon הינו מחזר שעון בודד, גודל הבלוק הינו 8 מילימ, ומדיניות הכתיבה הינה write back, write allocate. המטרון ממומש באמצעות שני זכרונות RAM: הזיכרון הראשון, DSRAM, הינו ברוחב 32 ביטים ובעומק 512 מילימ, ומכיל את הדטה השמור במתomon.

הזיכרון השני, TSRAM, מכיל 64 שורות, כאשר כל שורה מכילה עבור כל בלוק במתomon את התג ואת מצב פרוטוקול הקוهرנטיות MESI:

| 13:12   | 11:0 |
|---|------|
| MESI<br>(0: Invalid, 1: Shared, 2: Exclusive,<br>3: Modified) | Tag  |

בתחלת הריצה זכרונות ה- DSRAM ו- TSRAM מאופסים.

## זיכרון ראשי ו- MESI BUS

הגישה לנוטונים הינה למילימ בלבד (אין תמייה בביטים). מרחב כתובות הנוטונים הינו 21 ביטים, וגודל הזיכרון הראשי הינו 2 בחזקת 21 מילימ.

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

|            |         |  |
|------------|---------|--|
| bus_origid | 3 bits  | Originator of this transaction<br>0: core 0<br>1: core 1<br>2: core 2<br>3: core 3<br>4: main memory           |
| bus_cmd    | 2 bits  | 0: no command<br>1: BusRd<br>2: BusRdX<br>3: Flush   |
| bus_addr   | 21 bits | word address   |
| bus_data   | 32 bits | word data  |
| bus_shared | 1 bit   | Set to 1 when answering a BusRd transaction if any of the cores has the data in the cache, otherwise set to 0. |

בכל מחזור שעון על קווי הבס יש טרנסקציה בודדת. במידה ומספר מטמוניים רוצחים לפנות לבס, מתבצעת ארביטרציה הוגנת בשיטת Round-Robin (הליבה האחרונה שקיבלה גישה היא CUTT בעדיפות אחרונה). לא ניתן גישה לבס לlibva מסוימת כל עוד הבס תפוס או כל עוד טרנסקציה קודמת מסוג BusRd או BusRdX לא הסתיימה ע"י טרנסקציית Flush.

מי שיוצר את הטרנסקציה וכותב לקווי הבס במחזור שעון זה (ליבה או הזיכרון הראשי) מלא קוד בKO ה- bus\_origid.

הוראות BusRd ו- BusRdX מעבירות כפרמטר את כתובת המילה בזיכרון הראשי. במידה והדאטא עדכני בזיכרון הראשי, הזיכרון הראשי יחזיר את הבלוק שכולל את המילה באמצעות הוראות Flush. המילה הראשונה בבלוק תוחזר בהשניה של 16 מוחזורי שעון לאחר קבלת הוראת הקירהה, ושאר המילים בבלוק במחזורי שעון עוקבים. הדאטא יוחזר בקוווי ה- bus\_data, וכתובת המילה בזיכרון הראשי ב- .bus\_addr

עבור RdBus, במתן הוראה לבס, bus\_shared מאותחל לאפס. חלק ממנגנון זה Snooping כל מזמן בכל ליבה בודק האם יש פגיעה, ואם כן מעלה את קו ה- bus\_shared ל - 1. כאשר הדאטא חוזר, בהסתמך על קו ה- bus, מצב הפרוטוקול יקבע כ- Shared או Exclusive.

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

## **סט ההוראות וקידודם**

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

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

:האופקודות הנתמכים ע"י המעבד ומשמעות כל הוראה נתונים בטבלה הבאה

| Meaning                  | Name | Opcode Number |
|--------------------------|------|---------------|
| $R[rd] = R[rs] + R[rt]$  | add  | 0             |
| $R[rd] = R[rs] - R[rt]$  | sub  | 1             |
| $R[rd] = R[rs] \& R[rt]$ | and  | 2             |
| $R[rd] = R[rs]   R[rt]$  | or   | 3             |
| $R[rd] = R[rs] ^ R[rt]$  | xor  | 4             |
| $R[rd] = R[rs] * R[rt]$  | mul  | 5             |
| $R[rd] = R[rs] << R[rt]$ | sll  | 6             |

|  |      |    |
|--|------|----|
| R[rd] = R[rs] >> R[rt], arithmetic shift with sign extension                             | sra  | 7  |
| R[rd] = R[rs] >> R[rt], logical shift  | srl  | 8  |
| if (R[rs] == R[rt]) pc = R[rd][low bits 9:0]   | beq  | 9  |
| if (R[rs] != R[rt]) pc = R[rd] [low bits 9:0]  | bne  | 10 |
| if (R[rs] < R[rt]) pc = R[rd] [low bits 9:0]   | blt  | 11 |
| if (R[rs] > R[rt]) pc = R[rd] [low bits 9:0]   | bgt  | 12 |
| if (R[rs] <= R[rt]) pc = R[rd] [low bits 9:0]  | ble  | 13 |
| if (R[rs] >= R[rt]) pc = R[rd] [low bits 9:0]  | bge  | 14 |
| R[15] = next instruction address, pc = R[rd][9:0]  | jal  | 15 |
| R[rd] = MEM[R[rs]+R[rt]]   | lw   | 16 |
| MEM[R[rs]+R[rt]] = R[rd]   | sw   | 17 |
| Halt this core<br>Exit simulator when all cores reached halt and the pipelines are empty | halt | 20 |

## סימולטור

הסימולטור מסמלץ את צנורות הליבות, המטמוניים, והזיכרון הראשי. בתחילת הרצה כל ליבת מתחילה לזרז החל מ-0=PC. סיום הרצה ויציאה מהסימולטור מתבצע כאשר כל הליבות ביצעו את הוראת ה-HALT והצנורות התרוקנו.

הסימולטור יכתב בשפת C ויקומפל לתוכן command line application אשר מקבל 27 command line parameters לפי שורת הזרה 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, אז 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 הרצה. כל שורה בקובץ מכילה תוכן שורה בזיכרון ההוראות, החל מכתובת אפס, בפורמט של 8 ספרות הקסדצימליות. במידה ומספר השורות בקובץ קטן מ-1024, ההנחה הינה ששאר הזיכרון מעל הכתובת الأخيرة שאותחה בקובץ, מאופס. ניתן להניח שקובץ הקטל תקין.

הקובץ memin.txt הינו קובץ קלט בפורמט טקסט אשר מכיל את תוכן הזיכרון הראשי בתחילת הרצה. כל שורה בקובץ מכילה תוכן שורה בזיכרון, החל מכתובת אפס, בפורמט של 8 ספרות הקסדצימליות. במידה ומספר השורות בקובץ קטן מ-2 בחזקת 21 ההנחה הינה ששאר הזיכרון מעל הכתובת الأخيرة שאותחה בקובץ, מאופס. ניתן להניח שקובץ הקטל תקין.

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

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

הקבצים **core0trace.txt** – **core3trace.txt** הינם קבצי פלט, המכילים מעקב אחר מצב הצנרת. עבור כל ליבה, בכל מחזור שעון שלפחות אחד מהשלבים בcenret פעיל, תודפס שורת טסקט המציין איזו הוראה נמצאת בכל שלב בcenret, וכןן 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 בסיס דצימלי.  
השדות FETCH עד WB מכילים את ה- PC של הוראה שנמצאת בשלב המתאים בcenret ומודפסים ב- 3 ספרות הקסדצימליות. במידה ושלב הצנרת אינם פעילים (למשל בעת مليוי הצנרת או כאשר יש בוועית בcenret 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 במהלך זה), כאשר כל רגיסטר מודפס ב- 8 ספרות הקסדצימליות.

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

CYCLE bus\_origid bus\_cmd bus\_addr bus\_data bus\_shared

השדה CYCLE הינו מספר מחזור השעון ומודפס בסיס דצימלי. שאר השדות מודפסות בהקסדצימלי (שדות bus\_origid, bus\_shared, bus\_cmd, bus\_addr בספירה בודדת, bus\_data ב- 6 ספרות ו- bus\_addr ב- 8 ספרות).

הקבצים **dsram0.txt** – **dsram3.txt** הינם קבצי פלט, בהם פורמט כמו txt.memin, שכל אחד מהם מכיל את תוכן זיכרון ה- DSRAM של המטען בלביה המתאים בסיום הריצה, כאשר כל שורה מודפסת ב- 8 ספרות הקסדצימליות.

הקבצים **tsram0.txt** – **tsram3.txt** הינם קבצי פלט, בהם פורמט כמו txt.memin, שכל אחד מהם מכיל את תוכן זיכרון ה- TSRAM של המטען בלביה המתאים בסיום הריצה, כאשר כל שורה מודפסת ב- 8 ספרות הקסדצימליות.

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

| 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 זהות שלכם.
2. הפרויקט יכתב בשפת התוכנות סי. יש להקפיד שהיו הערות בתוך הקוד המסבירות את פועלתו.
3. יש להגיש את הקוד ב- windows. יש להגיש את קובץ ה- solution, ולוודא שהקוד מתקמפל ורץ, כך שניתן יהיה לבנות אותו ע"י לחיצה על solution build. יש להגיש גם את ספריית ה- build כולל קובץ ה- executable הבניין.
4. תוכניות בדיקה. הפרויקט שלכם יבדק בין השאר ע"י תוכניות בדיקה שלא תקבלו מראש, וגם ע"י תוכניות בדיקה שאתם כתבו באSEMBLY. יש להגיש שלוש תוכניות בדיקה:
  - א. תוכנית אשר רצה על 4 ליבות במקביל, כאשר כל ליבה מקדמת מונה בכתובות 0 בזיכרון הראשי (אשר מאותחל לאפס) 128 פעמים, אבל הקידום מתבצע בטור לפי סדר הליביות. קלומר: ליבה 0 מבצעת את הקידום הראשון, אח"כ ליבה 1 מבצעת את הקידום הבא, ועוד אז ליבה 2, ועוד ליבה 3, וחוזרים לライブ 0. בסיום הריצה יש לוודא כי

הערך בכתובת 0 (שאמור להיות 512) נכתב חזרה לזיכרון הראשי  
(באמצעות אילוץ conflict miss).

- ב. תוכנית המבצעת כפל של שתי מטריצות בגודל 16x16 (כל איבר במטריצה התוצאה הינו מכפלה סקלרית של שורה בעמודה), אשר רצאה על ליבנה בודדת, ליבנה מספר 0. ערכי המטריצה הראשונה נמצאים בכתובות 0 עד 0xFF בזיכרון הראשי, והמטריצה השנייה בכתובות 0x1FF עד 0x0. מטריצת התוצאה תיכתב לכתובות 0x200 עד 0x2FF. כל מטריצה מסודרת בזיכרון כמו בשפת סי: ערכי שורה ראשונה משמאלי לימין, ואז עוברים לשורה הבאה וכך הלאה. ניתן להניח שערכי המטריצות קטנים מספיק כך שלא יתרחש overflow.
- ג. בוצע כפל של אותן המטריצות כמו בתוכנית בדיקה ב', אבל על כל 4 הליבות במקביל, כאשר כל ליבנה מבצעת חלק מהחישוב. **ינתן ניקוד גבוה יותר לזמן ריצה כולל נמוך יותר.**

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

counter, mulserial, mulparallel

כל ספרייה תכיל 28 קבצים: את קובץ ההרצה sim.exe, את קבצי הקלט הדרושים להרצה, וכןן 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