BAR-ILAN UNIVERSITY (RA)

Faculty of Engineering

Ramat-Gan 52900, Israel

אוניברסיטת בר-אילן (ע"ר)

הפקולטה להנדסה

רמת-גן 52900

Tel: 03-5317722

engbi@mail.biu.ac.il

תכן לוגי

תשע''ט סמסטר קיץ מועד מיוחד

83-253

מרצה: פרופ' שמואל וימר

מתרגל: מר בנימין פרנקל

- **יש לקרוא היטב את ההוראות.**
- **משך הבחינה: שעתיים.** לא תינתן כל הארכה (למעט סטודנטים בעלי זכאות לתוספת זמן).
- חובה לענות על כל השאלות.
- סך הנקודות בבחינה הוא 120, אך הציון הסופי בבחינה לא יעלה על 100.
- יש לנמק ולהסביר את כל תשובותיכם.
- **כל חומר עזר מותר בשימוש**.
- **העבודה חייבת להיות עצמאית לחלוטין! אסור בתכלית האיסור להיעזר בשום אדם אחר! שימו-לב, כי אם יתעורר חשד לגבי טוהר הבחינה, נבחנים יידרשו להגן בעל-פה על תשובותיהם!**
- **<u>מומלץ מאד</u> להיות נוכח במצב וידאו במערכת ה- ZOOM בקישור שנשלח, וזאת על-מנת שאפשר יהיה לענות לשאלות שתתעוררנה במהלך הבחינה, בדומה למענה בזמן בחינה בכיתה. זו הדרך היחידה לקבלת מענה.**
- אין לחרוג בשום אופן מזמן הבחינה, ויש להעלות את התשובות למערכת ה- Moodle בחלון הזמן הנדרש. אי-העלאה בזמן תיחשב כמו אי-מסירת מחברת בבחינה רגילה.
- יש להקפיד על כתב יד קריא! ניתן להעלות תצלום ברור של כתב ידכם או להעלות קובץ word.

בהצלחה!

1. עליכם לאשר בכתב את ההתחייבות הבאה: אני מתחייב/ת לשמור על טוהר הבחינה, ומצהיר/ה שלא אשתמש בעזרתו של שום אדם אחר במהלך פתרון הבחינה. (0 נקודות)

השאלה שלהלן דנה בכפל מספרים במעבד MIPS. מניחים שמדובר רק בהכפלת מספרים אי שליליים המיוצגים בשיטת המשלים ל 2. כמו-כן, ניתן להתעלם לחלוטין מזמני setup ו- hold ומההשהיות הפנימיות של אוגרים ככל שהדבר נדרש.

נתון מעבד MIPS מצונר בן 32 סיביות, במימוש וביצועים שלהלן:

- תדר שעון 100MHz.
- בעל יחידות forwarding ככל שנדרש.
- בדיקת קפיצה מותנית נעשית בשלב ה- ID בשילוב ריקון פקודה אם נדרש.
- פקודת קפיצה מחלטת (jump) מסתיימת בשלב ה- ID.

המעבד הנ"ל נדרש לבצע קוד המכיל הכפלות מרובות.

2. מהם הערכים המרביים האפשריים לכופל ולנכפל, כאלו שיבטיחו שתוצאת הכפל תמיד תהיה חוקית? חשבו במדויק והסבירו את תשובתכם. (8 נקודות)

מאחר ומדובר בארכיטקטורת 32 סיביות וייצוג משלים ל- 2, המספר החיובי המרבי הניתן לייצוג הינו: $2^{31}-1$, ועל-כן התשובה הינה: $\lfloor\sqrt{2^{31}-1}\rfloor = 46{,}340$.

חברת המעבדים "המחשב המהיר" החליטה לבחון שתי חלופות להשגת ביצועים מיטביים. האחת, פתרון בחומרה, ע"י הוספת מכפל ל- ALU, והשנייה ע"י פתרון תוכנתי של כתיבת קוד לאלגוריתם כפל.

ידוע שמכפל צירופי הינו מעגל גדול ומורכב. ראובן, מהנדס נמרץ בוגר BIU בהנדסת חשמל, הצליח לאחר מאמצים מרובים לתכנן מכפל, ובצירוף כל הלוגיקה הנדרשת ההשהיה של פעולת כפל הינה 50nSec. בכדי לתמוך בכפל נוספה פקודה בפורמט ALU: mult $s2, $s0, $s1.

3. מהי השפעת תוספת המכפל על זמני הריצה של תוכניות שאינן מכילות פעולות כפל (ללא שינוי של עומק ה- pipeline)? חשבו במדויק והסבירו את תשובתכם. (8 נקודות)

הוספת המכפל ל- ALU מחייבת להאט את השעון לתדר $\mathbf{1/(50\times10^{-9}) = 20MHz}$. על-כן, זמני הריצה של תכניות שאינן מכילות הכפלות תארכנה פי חמש זמן.

שמעון, גם הוא מהנדס נמרץ בוגר BIU בהנדסת מחשבים, הציע פתרון תוכנה ע"י קוד המממש כפל ארוך (כזה הנלמד בבית הספר היסודי) המתואר להלן. הקוד מניח שהנכפל והכופל מצויים באוגרים s0 ו- s1 בהתאמה. הפקודה li הינה load immediate והפקודה addi הינה add immediate.

```
1           li $t1, 0 # Counter for loop.
2           li $s3, 1 # Mask for extracting bit!
3           li $s2, 0 # Result's going to be in s2!
4   multiply:
5           beq $t1, 16, exit
6           and $t0, $s1, $s3
```

```
 7          sll $s3, $s3, 1
 8          beq $t0, 0, increase
 9          add $s2, $s2, $s0
10    increase:
11          addi $t1, $t1, 1
12          j multiply
13          sll $s0, $s0, 1
14    exit:
15          add $a0, $s2, $zero
```

4. מהי הסיבה שבגללה החליט שמעון לרשום את הפקודות בשורות 12 ו- 13 כפי שהן ולא בסדר הפוך? (9 נקודות)

הפקודה בכתובת multiply של פקודת קפיצה מחלטת (jump) תתבצע רק כעבור שני מחזורי שעון מכניסת פקודת הקפיצה לצינור, זמן שבו כבר נכנסה פקודה נוספת. על-כן, כדי שלא לבזבז מחזור שעון כדאי שזו תהיה פקודה שצריכה להתבצע בכל מקרה. לו הסדר היה הפוך הייתה נדרשת פקודת nop אחרי ה- jump, או לחליפין מנגנון חומרה הדואג להשמדת הפקודה המידית בקוד אחרי ה jump. בכל מקרה היה אובדן מחזור שעון בלולאה. (הערה: גם לו סדר הפקודות 12 ו-13 היה הפוך, במקרה הזה לא הייתה נוצרת בעיה לוגית, שכן פקודת ה- add משורה 15 יכולה להתבצע גם "בטעות", והיא איננה פוגעת בנכונות התוכנית. מה שכן, היה נגרם נזק לביצועי התוכנית, שכן כל לולאה הייתה אורכת מחזור שעון נוסף).

5. מנהל החברה החליט לבדוק את השפעת הפתרון האלגוריתמי על זמני הריצה של תוכניות ולצורך כך הגדיר שני test bench. הראשון מכיל תוכניות שאינן כוללות הכפלות, ואלו בשני 4% מהפעולות הינן כפל. מניחים CPI=1 עבור כל התוכניות. פי כמה יתארך זמן הריצה של ה-test bench השני לעומת הראשון? הסבירו ונמקו כל צעד בחישוב. (10 נקודות)

ראשית, יש לחשב את מספר מחזורי השעון הנדרש להכפלת שני מספרים. בקוד הנ"ל ישנן 3 פקודות אתחול. לאחריהן לולאה בת 8 פקודות החוזרת 16 פעמים. שימו-לב שפקודת הקפיצה המותנית beq $t0, 0, inc אינה גורמת לאבדן מחזור שעון בין שהקפיצה מתרחשת ובין שלא. לבסוף, ישנה היציאה מהלולאה שמצריכה 3 מחזורי שעון בגין ביצוע פקודת הקפיצה המותנית בשורה 5, השמדת פקודת ה- and שאחריה ופקודת העתקת התוצאה לאוגר a0. בסה"כ מדובר ב-

$$\mathbf{3 + 16 \times 8 + 3 = 134}$$

מחזורי שעון. על כן, קוד המכיל 4% פעולות כפל יארך פי

$$\mathbf{0.96 \times 1 + 0.04 \times 134 = 6.32}$$

מחזורי שעון מקוד ללא הכפלות.

6. מה צריך להיות אחוז פעולות הכפל בקוד כך שזמני הריצה (במונחי שניות) בפתרונו של ראובן יהיו קצרים יותר מאלה של שמעון? רשמו את הפתרון באופן מלא. (10 נקודות)

יהי $0 \le x \le 1.0$ החלק היחסי של פעולות הכפל. זמן מחזור השעון בפתרון של ראובן גדל פי 5 לעומת זמן מחזור השעון של המעבד הרגיל. נייחס ערך 5t ו- t לזמני מחזורי השעון בהתאמה. צריך להתקיים:

$$5t < (1.0 - x) \times t + x \times 134 \times t \Rightarrow x > 4/133 = 0.030075$$

כלומר, יותר מ 3.0075% מהפקודות צריכות להיות כפל.

7. מה תהיה השפעת סילוק יחידות ה- forwarding על הקוד הנ"ל? אם הדבר גורם לבעיות כלשהן בהרצת הקוד יש לפתור כל בעיה ובעיה באופן מיטבי.
דונו בשני המקרים הבאים:

- כתיבת ערך לרגיסטר ב- Register File וקריאת הערך העדכני מאותו רגיסטר יכולות להתרחש באותו מחזור שעון.
- כתיבת ערך לרגיסטר ב- Register File וקריאת הערך העדכני מאותו רגיסטר אינן יכולות להתרחש באותו מחזור שעון, אלא רק בשני מחזורי שעון נפרדים. (10 נקודות)

יש לבדוק האם והיכן מתרחשים data hazards. נזכור שחייב להיות מרחק של לפחות שתי פקודות (אם ניתן לכתוב ולקרוא ל- RF באותו מחזור שעון) או שלוש פקודות (אם לא ניתן לכתוב ולקרוא ל- RF באותו מחזור שעון) כדי שתוצאה הנרשמת באוגר בשלב ה- WB תהיה זמינה לפקודה הנמצאת בשלב ה- ID. מאחר והאוגר t0 הנרשם בפקודה שבשורה 6 נדרש בפקודה שבשורה 8, תידרש הכנסת פקודת nop ביניהן (אם ניתן לכתוב ולקרוא באותו מחזור שעון) או שתי פקודות nop (אם לא ניתן לכתוב ולקרוא באותו מחזור שעון).

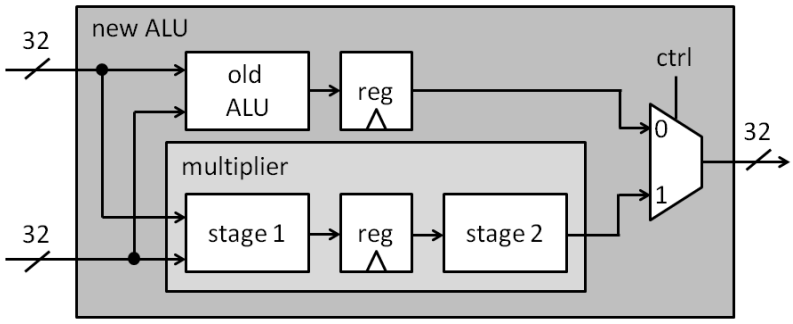
בנוסף, אם לא ניתן לכתוב ולקרוא באותו מחזור שעון, הרי שישנו data hazard בין הפקודה שבשורה מספר 5 לבין הפקודה שבשורה מספר 1, וכן בין הפקודה שבשורה מספר 6 לבין הפקודה שבשורה מספר 2. באופן דומה, ישנו data hazard בין הפקודה שבשורה מספר 5 לבין הפקודה שבשורה מספר 11 (באיטרציה הקודמת של הלולאה). כדי לפתור את שלוש הבעיות הנ"ל, ניתן כמובן להכניס פקודת nop לפני הפקודה שבשורה מספר 5, אבל זה יגדיל את מספר מחזורי השעון בלולאה בהתאם.

פתרון יעיל יותר הינו בשינוי סדר הפקודות. אם ניתן לקרוא ולכתוב באותו מחזור שעון, די היה להקדים את ביצוע פקודת ה- addi הרשומה בשורה מספר 11 כך שתתבצע לאחר זו הרשומה בשורה מספר 7, וכך אין שום צורך בהוספת nop. אם אין אפשרות לקרוא ולכתוב באותו מחזור שעון, הרי שניתן לבצע שינוי בסדר הפקודות באופן הבא:

```
1           li $s3, 1 # Mask for extracting bit!
2           li $t1, 0 # Counter for loop.
3           li $s2, 0 # Result's going to be in s2!
4           nop
5   multiply:
6           and $t0, $s1, $s3
7           beq $t1, 16, exit
8           sll $s3, $s3, 1
9           addi $t1, $t1, 1
10          beq $t0, 0, increase
11          add $s2, $s2, $s0
12  increase:
13          j multiply
14          sll $s0, $s0, 1
15  exit:
16          add $a0, $s2, $zero
```

באופן הזה, **כל** ה- data hazards נפתרו, וזאת במחיר של הוספת nop אחד ויחיד, המתבצע פעם אחת בלבד (מכיוון שהוא מחוץ ללולאה). אמנם, כאשר תתרחש הקפיצה שבשורה מספר 7 (על-פי סדר הפקודות החדש) הדבר יגרום לביצוע מיותר של פקודת ה- and שלפניה, אבל אין בכך שום בעיה.

כדי להתגבר על בעיית ההשהיה הארוכה של המכפל הצירופי גילה ראובן שניתן לחלק את פעולת הכפל המתבצעת במעגל החומרה לשני שלבים טוריים, כך שההשהיות הנדרשות בשניהם שוות. בהתאם לכך, הציע תכן חדש ל- ALU הנראה בתרשים הבא:



8. מה תפקידו של ה- MUX ביציאת ה- new ALU? (7 נקודות)

תפקיד ה- MUX לבחור בין תוצאת פעולת כפל לבין הפעולות האחרות שמבצע ה- ALU.

9. מדוע נדרש אוגר המחובר בטור ל- old ALU? (8 נקודות)

הדבר נדרש על-מנת לאזן את עומק הצינור כך שכל הפעולות הנדרשות בשלב ה- EXE ימשכו שני מחזורי שעון, הפקודות השונות תתבצענה בסדר הנכון והתוצאות יכתבו ל- register file בסדר הנכון.

10. בהתעלם מאבדן מחזורי שעון עקב hazards שונים, האם לתכן הנ"ל ישנה השפעה עקרונית על ה- CPI? נמקו את תשובתכם באופן מפורט. (9 נקודות)

לדבר אין השפעה עקרונית על ה- CPI משום שעדיין בכל מחזור שעון מסתיימת פקודה.

11. מה צריך להיות אחוז פעולות הכפל בתכן הנ"ל כך שזמני הריצה (במונחי שניות) בפתרונו של ראובן יהיו קצרים יותר מאלה של שמעון? רשמו את הפתרון באופן מלא. (10 נקודות)

זמן מחזור השעון בפתרון המכפל החדש של ראובן הינו 25nSec כלומר פי 2.5 לעומת זמן מחזור השעון של המעבד הרגיל. נייחס ערך 2.5t ו- t לזמני מחזורי השעון בהתאמה. צריך להתקיים:

$$\mathbf{2.5t} < (\mathbf{1.0} - \boldsymbol{x}) \times \mathbf{t} + \boldsymbol{x} \times \mathbf{134} \times \mathbf{t} \Rightarrow \boldsymbol{x} > \mathbf{1.5/133} = \mathbf{0.01128}$$

כלומר, יותר מ- 1.128%

12. בהנחה שאין כל שינויים ותוספות במנגנוני ה- forwarding הקיימים בתכן המקורי, האם לתכן החדש של ה- ALU תהיה השפעה כלשהי על data hazards? כיצד ישפיע הדבר על קוד ה- assembly של תוכניות כלשהן? נמקו את תשובתכם באופן מפורט. (9 נקודות)

מאחר וב- ALU החדש פעולה אורכת שני מחזורי שעון, המרחק בין שלב ה WB שבו נכתבת תוצאה ל- register file לבין קריאת התוצאה ממנו גדל משניים לשלשה מחזורי שעון. הדבר עשוי רק להגדיל את כמות ה data hazards. בהעדר מנגנוני forwarding נוספים תידרשנה פקודות nop נוספות בקוד, ועקב כך זמן הביצוע (במחזורי שעון) יתארך.

13. אם נדרש להריץ את אלגוריתם הכפל הנ"ל בארכיטקטורת multi-cycle, אילו שינויים מחויבים להתבצע בקוד? ציינו במפורש כל שינוי ושינוי. (9 נקודות)

בארכיטקטורת multi-cycle נדרש להחליף את הסדר בין פקודות 12 ו- 13, שכן פקודת ה- sll צריכה להתבצע כחלק מהלולאה. בארכיטקטורת pipeline ניצלנו את ה- delay slot על מנת לבצע פקודה שצריכה להתבצע בכל מקרה. לעומת זאת, בארכיטקטורת multi-cycle כל פקודה מתחילה להתבצע רק לאחר שהפקודה הקודמת הסתיימה, ולכן, אם נשאיר את הפקודות בסדר הנוכחי, פקודת ה- sll משורה 13 לא תתבצע בתוך הלולאה, מה שיפגע בנכונות אלגוריתם הכפל.

נתון כי פקודת li ב- multi-cycle-MIPS ממומשת באופן היעיל ביותר מבחינת מספר מחזורי השעון הנדרשים לביצוע הפקודה.

14. מהי ההאצה המרבית והמזערית היכולה להתקבל בביצוע אלגוריתם הכפל בארכיטקטורה מצונרת לעומת MIPS הממומש בארכיטקטורת multi-cycle? חשבו את תשובתכם באופן מפורט. <u>רמז:</u> ההאצה תלויה בערכו של הכופל, אם כן, מצאו תחילה איזה כופל מוביל להאצה מרבית ואיזה כופל מוביל להאצה מזערית. (13 נקודות)

ראשית, יש למצוא את מספר מחזורי השעון הנדרש ב multi-cycle MIPS. לשם כך צריך לדעת כמה מחזורי שעון נדרשים לכל פקודה.
פקודת li דורשת 3 מחזורי שעון (בניגוד ל- load רגיל הדורש 5 מחזורים). קפיצה מותנית וקפיצה מוחלטת 3 מחזורים. פקודות ALU כולל addi דורשות 4 מחזורים.
שלוש הפקודות הראשונות לפני הלולאה ידרשו סה"כ 9 מחזורים.
בתוך הלולאה ישנן שלוש פקודות קפיצה הדורשים 9 מחזורים. מספר פקודות ה- ALU וה- addi בתוך הלולאה תלוי בערך הסיבית המתאימה בכופל. אם הסיבית ה- i בכופל היא '0', הרי שבאיטרציה ה- i הקפיצה המותנית בשורה 8 תילקח, ולא תתבצע פקודת ה- add המופיעה בשורה 9. במקרה הזה, ביצוע הלולאה דורש 16 מחזורי שעון לארבע פקודות ה- ALU וה- addi ועוד 9 מחזורים לשלש הקפיצות, סה"כ 25 מחזורי שעון לביצוע הלולאה באיטרציה ה- i.
אם הסיבית ה- i בכופל היא '1', הרי שבאיטרציה ה- i הקפיצה המותנית בשורה 8 לא תילקח, ופקודת ה- add המופיעה בשורה 9 כן תתבצע. במקרה הזה, ביצוע הלולאה דורש 20 מחזורי שעון לחמש פקודות ה- ALU וה- addi ועוד 9 מחזורים לשלש הקפיצות, סה"כ 29 מחזורי שעון לביצוע הלולאה באיטרציה ה- i.
בכניסה הנוספת ללולאה מתבצעת הפקודה שבשורה 5, ומיד אחריה הפקודה שבשורה 15, סה"כ 7 מחזורי שעון נוספים.
כופל אשר יוביל להאצה מרבית בביצוע אלגוריתם הכפל בארכיטקטורה מצונרת לעומת MIPS הממומש בארכיטקטורת multi-cycle הוא כופל אשר יגרום למספר מחזורי שעון מירבי בביצוע האלגוריתם בארכיטקטורת multi-cycle, והוא כופל אשר כל הסיביות שלו הן '1'. במקרה כזה, ידרשו בסה"כ:

$$9 + 16 \times 29 + 7 = 480$$

מחזורי שעון, וזאת לעומת 134 ב- MIPS מצונר.
אם-כן, ההאצה המרבית הינה: $480/134 = 3.582$.
כופל אשר יוביל להאצה מזערית בביצוע אלגוריתם הכפל בארכיטקטורה מצונרת לעומת MIPS הממומש בארכיטקטורת multi-cycle הוא כופל אשר יגרום למספר מחזורי שעון מזערי בביצוע האלגוריתם בארכיטקטורת multi-cycle, והוא כופל אשר כל הסיביות שלו הן '0'. במקרה כזה, ידרשו בסה"כ:

$$9 + 16 \times 25 + 7 = 416$$

מחזורי שעון, וזאת לעומת 134 ב- MIPS מצונר.
אם-כן, ההאצה המזערית הינה: $416/134 = 3.104$.