Archlab – Lab 2

Gal Kashi – 204572861

Chen Eilon – 201617032

שאלה 1

1. הערכים שיתקבלו:

$$r[2] = 12$$
$$r[3] = 45$$
$$r[4] = 57$$
$$r[5] = -12$$

2. הקוד שגוי משתי סיבות:
   - מתבצעת קריאה מ-sprn. בשלב ריצת הפונקציה run ניתן רק לכתוב ל-sprn.
   - מתבצעת כתיבה ל-spro. בשלב ריצת הפונקציה ()run ניתן רק לקרוא מ-spro.

שאלה 2

1. בכל הוראה נעבור ב-6 מצבים. כל אחד מהמצבים האלה לוקח מחזור שעון אחד. מכיוון שאין צנרת, אין חפיפה בביצוע ההוראות. לכן ביצוע n הוראות יקח 6n מחזורי שעון.

2. ניתן לתמוך בקריאות עוקבות לזכרון אם נוסיף יחידת SRAM נוספת וזהה לזו הקיימת, ונדאג לשמור על עקביות בין שתי היחידות. כך אם שלחנו בקשה לקריאה במחזור n, נוכל לקרוא את המילה שביקשנו מה-SRAM הראשון במחזור n+1 ובמקביל לשלוח בקשת קריאה נוספת, ולקרוא את המילה השניה שביקשנו מה-SRAM השני במחזור n+2.

3. יתרונות:
   - זולה יותר.
   - פחות נוטה לשגיאות.

   חסרונות:
   - פחות יעילה.

שאלה 3

הקובץ מצורף (בתיקיה sp).

שאלה 4

הקבצים מצורפים (בתיקיה mult).

שאלה 5

הקבצים מצורפים (בתיקיה fibo).

שאלה 6

יחידת ה-DMA מורכבת מ-5 רגיסטרים וממכונת מצבים.

הרגיסטרים של יחידת ה-DMA:

dma_state[1:0] – המצב של מכונת המצבים.

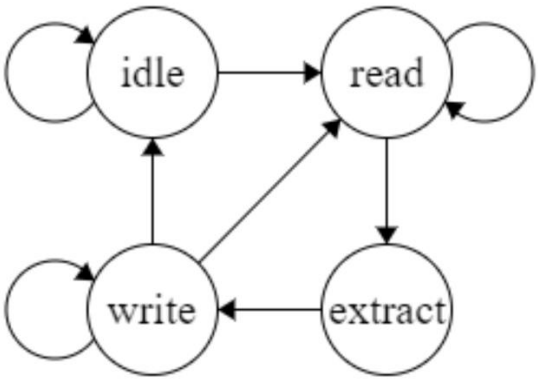
dma_src[15:0] – כתובת המקור.

dma_dst[15:0] – כתובת היעד.

dma_counter[15:0] – מונה עבור מספר ההעתקות. סופר אחורה עד ל-0.

dma_reg[31:0] – רגיסטר עבודה אליו נקראים הנתונים מהמקור וממנו נכתבים הנתונים ליעד.

מבנה מכונת המצבים של יחידת ה-DMA:



1. **idle** – היחידה מחכה לפעולת העתקה. כשמתקבלת פעולה כזו, היא עוברת למצב read. אחרת, נשארת ב-idle.
2. **read** –אם התוכנית הראשית צריכה לגשת לזכרון במחזור הנוכחי או בזה שאחריו (או אם לא ניתן לקבוע בוודאות), היחידה נשארת במצב read מבלי לעשות דבר. אחרת, היא שולחת בקשת קריאה ל-SRAM מהכתובת השמורה ב-dma_src ועוברת למצב extract.
3. **extract** – היחידה טוענת את המילה שנקראה במחזור הקודם ל-dma_reg.
4. **write** – אם התוכנית הראשית צריכה לגשת לזכרון במחזור הנוכחי, היחידה נשארת במצב write מבלי לעשות דבר. אחרת, היא שומרת את המילה ב-dma_reg בכתובת השמורה ב-dma_dst, מקדמת את הכתובות dma_src, dma_dst, ומחסירה 1 מהמונה dma_counter. אם זוהי ההעתקה האחרונה לפעולה זו, היא עוברת למצב idle. אחרת, היא עוברת למצב read כדי להתחיל את ההעתקה הבאה.

על מנת להשתמש ביחידת ה-DMA הוספנו 2 הוראות חדשות ל-IS:

1. DMA (30) – מעתיק R[src0] מילים מהמקור החל מכתובת R[src1] ליעד החל מכתובת R[dst].

2. (31) POL – קובע את הערך של R[dst] להיות 1 אם קיימת פעולת DMA פעילה. אחרת, קובעת את הערך להיות 0. באחריות המפתח לבדוק שלא קיימת פעולת DMA פעילה לפני קריאה נוספת ל-DMA, גישה לזכרון שעליו רצה הפעולה, או סיום התוכנית.

קוד המקור מצורף (בתיקיה dma).

<u>שאלה 7</u>

קוד הבדיקה:

```
ADD, 2, 1, 0, 200   // 0: R2 = 200
ADD, 3, 1, 0, 500   // 1: R3 = 500
DMA, 3, 1, 2, 100   // 2: Copy MEM[R2:R2+100] to MEM[R3:R3+100]
ADD, 2, 1, 0, 30    // 3: R2 = 30
ADD, 3, 1, 0, 1     // 4: R3 = 1
ADD, 4, 1, 0, 8     // 5: R4 = 8
JEQ, 0, 3, 4, 14    // 6: PC = 14 if R3 == R4
LD,  5, 0, 2, 0     // 7: R5 = MEM[R2]
ADD, 2, 2, 1, 1     // 8: R2 = R2 + 1
LD,  6, 0, 2, 0     // 9: R6 = MEM[R2]
ADD, 6, 6, 5, 0     // 10: R6 = R6 + R5
ST,  0, 6, 2, 0     // 11: MEM[R2] = R6
ADD, 3, 3, 1, 1     // 12: R3 = R3 + 1
JEQ, 0, 0, 0, 6     // 13: PC = 6
POL, 2, 0, 0, 0     // 14: R2 = 1 if DMA is running, else 0
JNE, 0, 2, 0, 14    // 15: PC = 14 if R2 != 0
ADD, 2, 1, 0, 200   // 16: R2 = 200
ADD, 3, 1, 0, 500   // 17: R3 = 500
ADD, 4, 1, 0, 600   // 18: R4 = 600
JEQ, 0, 3, 4, 27    // 19: PC = 27 if R3 == R4
LD,  5, 0, 2, 0     // 20: R5 = MEM[R2]
LD,  6, 0, 3, 0     // 21: R6 = MEM[R3]
ADD, 2, 2, 1, 1     // 22: R2 = R2 + 1
ADD, 3, 3, 1, 1     // 23: R3 = R3 + 1
JEQ, 0, 5, 6, 19    // 24: PC = 19 if R5 == R6
ADD, 2, 1, 0, 0     // 25: R2 = 0
HLT, 0, 0, 0, 0     // 26: HALT
ADD, 2, 1, 0, 1     // 27: R2 = 1
HLT, 0, 0, 0, 0     // 28: HALT
```

הקוד מבוסס על תוכנית הדוגמה, מכיוון שהיא מכילה מספר קריאות לזכרון. ניתן לראות בקבצי הפלט כי התוכנית ביצעה את הפעולה המקורית שלה (חישוב סכום) במקביל להעתקת הבלוק, וכמו כן גם ההשוואה בסוף התוכנית הצליחה.

הקבצים הדרושים מצורפים (בתיקיה dma).