**Win13b**

1. נסמן ב- את הפונקציה שמחזירה 1 אמ"ם  ראשוני.  
   ROM מס' 1 יקבל את  ויחשב 4 פונקציות:  .  
   ROM מס' 2 יקבל את 4 הביטים הנ"ל ובנוסף יקבל את  ובהתאם להם יבחר איזה מארבעת הביטים שהגיעו מ-ROM מס' 1 לקחת.  
     
   סה"כ השתמשנו ב-2 ROMים, המסלול הכי ארוך עובר דרך שניהם ולכן מכיוון שהשהיית כל ROM הינה 10, ההשהיה של המערכת היא 20.
2. וואלה לא הבנתי מה כוונת המשורר בשאלה.
3. מצורפת תמונה...
4. נפתח נוסחה ל- ונקבל כי  .
5. 1. כל מכונה ניתן לקחת ולהוסיף שני רגיסטרים בכניסתה, זה ייצור מכונה זהה שמפגרת אחריה בשני מחזורים.
   2. המכונה שלנו היא מסוג מור ולכן קיימת מכונה שמקדימה אותה במחזור אחד. המכונה הנ"ל יכולה להיות מסוג מילי או מור. אם היא מסוג מילי אז אין מכונה שמקדימה אותה ולכן במקרה זה אין מכונה שמקדימה את המכונה המקורית בשני מחזורים. במקרה שמכונה זו היא מסוג מור קיימת מכונה שמקדימה אותה במחזור אחד, מכונה זו בעצם מקדימה את המכונה המקורית בשני מחזורים. לכן במקרה זה כן קיימת מכונה כנ"ל.  
      לכן התשובה – **אולי**.
   3. נראה שיש טעות בפיתרון הרשמי...  
      במקרה שמסלול הנתונים מכיל מסלול צירופי, שימוש בבקר שממומש ע"י ROM יגרום למעגל אך במקרה שאין במסלול הנתונים מסלול צירופי שימוש ב-ROM לא ייצור מסלול צירופי, אם שאר תנאי המשטר הדינאמי מתקיימים אז נשמר המשטר הדינאמי עבור המערכת.  
      לכן לדעתי התשובה פה צריכה להיות – **אולי**.
   4. האמת עכשיו כשאני חושב על זה יש מצב שהתכוונו לשאול את השאלות לגבי האם מסלול הנתונים יכול להיות ממומש ע"י ROM והאם הלוגיקות הצירופיות של מסלול הנתונים יכולות להיות ממומשות ע"י ROMים (במקום הבקר). אחרת זה נראה פחות הגיוני...  
      במקרה שזו אכן הכוונה: התשובה לסעיף הקודם היא שנוצר לנו מעגל כי הROM צירופי מכל כניסה לכל יציאה וכי קיים מסלול צירופי בבקר. התשובה לסעיף זה היא שמכיוון שלא קיים מסלול צירופי מכניסה ליציאה בבקר לא ניצור מעגל צירופי כשנחבר אותו למסלול הנתונים אז נוכל למממש ככה את המערכת מבלי שהיא תפר את המשטר הדינאמי.
6. המסלול הכי ארוך הוא 8. לכן מחזור שעון הוא 8. נתון שהחישוב לוקח 100 מחזורי שעון ולכן זמן החישוב יהיה 8\*100=800.  
   אם נבצע חתכים ככה שכל הקשתות בכיוון אחד נוכל להגיע תוך 7 חתכים למצב בו כל יחידה מבודדת מיחידות אחרות.  
   ביצענו 7 חתכים שרק הוסיפו רגיסטרים לכל המסלולים ולכן עיכבנו את החישוב ב-7 מחזורי שעון והוא ייקח עכשיו 107 מחזורים במקום 100. אבל הקטנו את זמן המחזור ל-1 ולכן זמן החישוב עתה יהיה 107\*1=107.  
     
   איך ידענו שצריכים לבצע רק 7 חתכים בדיוק? שתי אופציות:  
   א. ניסוי וטעייה (פחות מומלץ ולא ניתן להוכחה אבל עובד ;) ).  
   ב. אם נסתכל על המערכת נשים לב שכל מסלול מכניסה ליציאה בין אם צירופי ובין אם לא הוא באורך של 8 יחידות. קיים מסלול כזה שעליו 0 רגיסטרים בין היחידות ולכן נצטרך להוסיף 7 רגיסטרים למסלול הנ"ל (קיבלנו חסם תחתון- 7). מכיוון שאין מסלול שדורש הוספה של יותר מ-7 רגיסטרים, קיבלנו חסם עליון על מספר החתכים שצריכים לעשות (7).  
   (את 7 החתכים נעשה ככה שיתפרו את כל המסלולים).
7. כל גישה לזיכרון לוקחת עכשיו 2 מחזורים במקום 1 ולכן על כל גישה לזיכרון פקודה תארך מחזור שעון אחד יותר ממה שארכה עד עכשיו.  
   התקבלה התשובה 3 בפקודת J כנראה מכיוון שניתן לאחד את עדכון ה-PC (המחזור האחרון של J) עם שלב הDECODE ובעצם לחסוך מחזור.
8. 1. הדבר היחיד שמבדיל בין המערכת שלנו לפייפליין טהור (או פייפליין בכלל) זה שאין על כל מסלול אותו מספר של רגיסטרים, את זה ניתן כמובן לתקן ע"י הוספה של 3 רגיסטרים לאחר פייפליין A (לקו a).
   2. נרצה להפריד את לוגיקה L מ-A ומ-B ולכן נוסיף על הקווים a ו-b רגיסטר נוסף.  
      סה"כ הוספנו 4 ל-a. 1 ל-b. ב-c לא נגענו.
9. מה שדרוש בשביל לבצע את הפקודה זה פשוט לחבר את PC לקבוע K שיקודד בתור IM ולכתוב את התוצאה ל-PC. לא נדרשת תוספת של שום לוגיקה נוספת למסלול הנתונים ואת החיבור וכתיבתו ניתן לבצע במחזור השלישי לאחר FETCH ו- DECODE ולכן הפקודה תיקח 3 מחזורי שעון.
10. נשים לב מה החישוב שרוצים לבצע פה.  
    יש שני שלבים:  
    1. סכימת  מספרים, בעזרת CSA תוך  מחזורי שעון נגיע לשני מספרים שסכומם שווה לסכום  המספרים הללו. שני המספרים הללו יהיו באורך של  ביטים לכל היותר וחיבור שלהם ייקח לנו .  
    לכן שלב זה ייקח סה"כ- .  
    2. חלוקת התוצאה משלב 1 ב-, זה שקול למחיקת  הביטים הימניים של המספר וזה לוקח .  
      
    לכן סה"כ החישוב לקח-.