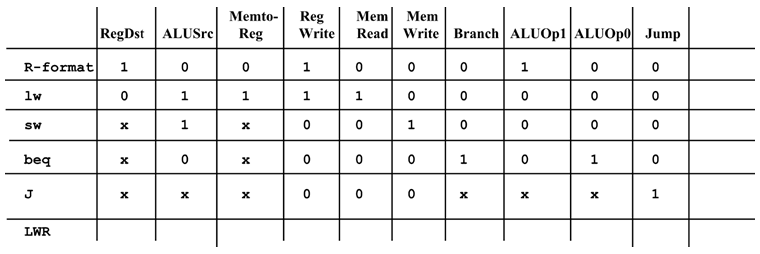
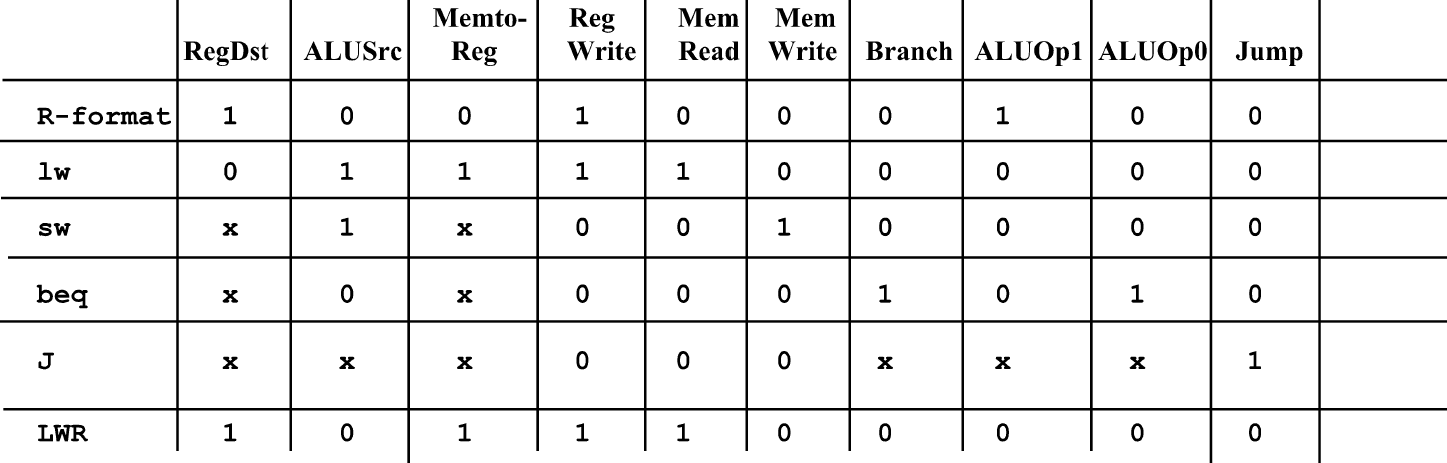
**שאלה א' (35 נקודות)**

נתון מעבד MIPS בעל מחזור יחיד כמופיע בתרשימים 1, 2 ו 3 (שלשתם זההים). בכל השאלות הבאות נדרש להרחיב את סט הפקודות וזאת ע"י שינויים מזעריים ככל האפשר במעבד הקיים. במדה ונדרש הוסיפו לתרשימים המתאימים משאבי חומרה ואותות בקרה, וקיבעו את ערכיהם של אותות הבקרה הקיימים ונוספים במדה שנדרש בטבלאות המתאימות. הסבירו באופן מלא ומדוייק את פתרונכם.

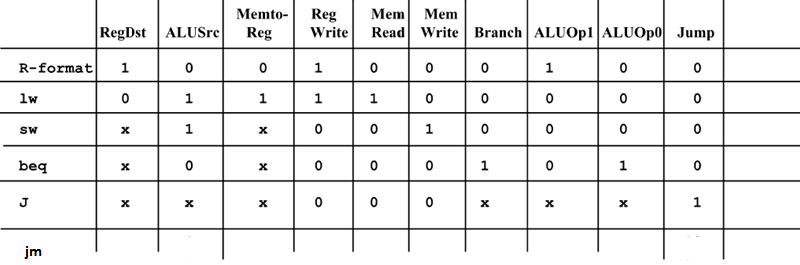
1. (10 נק') נדרש להוסיף תמיכה בפקודה חדשה המשתמשת ב R-format.הפקודה מחשבת כתובת אפקטיבית ע"י סיכום הרגיסטרים $rs ו $rt .



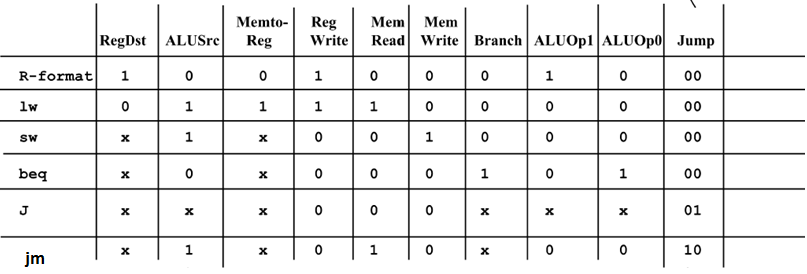
**תשובה**: לא נדרשת שום תוספת חומרה שהיא ושום אותות בקרה חדשים (תרשים 1 נשאר ללא שינוי). קביעת אותות הבקרה הקיימים עבור פקודת LWR מופיעה להלן.



1. (10 נק') נדרש להוסיף תמיכה בפקודה קפיצה חדשה המשתמשת ב I-format.הפקודה החדשה קוראת מהזכרון את הערך המצוי בכתובת האפקטיבית המוגדרת בפקודה וגורמת לתכנית המבוצעת לקפוץ לכתובת הנ"ל.



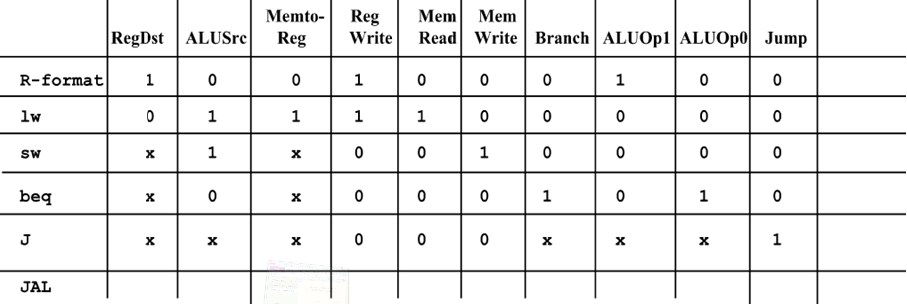
**תשובה**: נדרשת הרחבת ה MUX המשמש קפיצות בכניסה שלישית, דבר המחייב שתי סיביות באות הבקרה Jump. את הכניסה הנוספת של ה MUX מחברים ליציאת זכרון הנתונים.



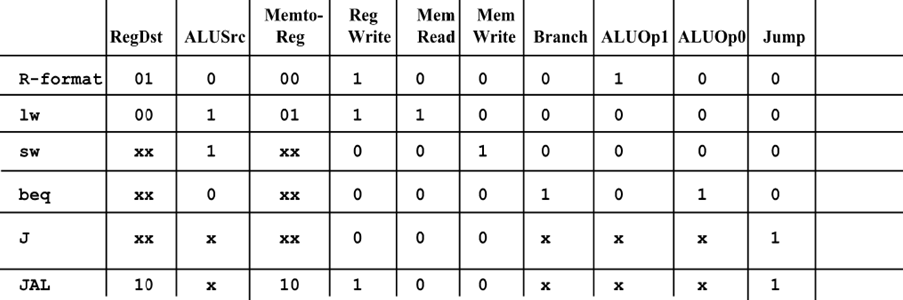
1. (2 נק') איזה יתרון יש לפקודה על פניjump רגיל?

**תשובה**: יתרונה של הפקודה בכך שהיא מאפשרת קפיצה לכל כתובת שהיא, בעוד ש jump רגיל מספק רק 28 סיביות, וארבחת ה MSB מוכתבים ע"י ה PC.

1. (10 נק') נדרש להוסיף תמיכה בפקודה קפיצה חדשה (jump and link) המשתמשת לקפיצה לסברוטינה, כך שבסיומה ידוע לאן צריך לחזור בתכנית הקוראת בכדי להמשיך בבצוע התכנית. כתובת החזרה נשמרת ב $ra ($31).

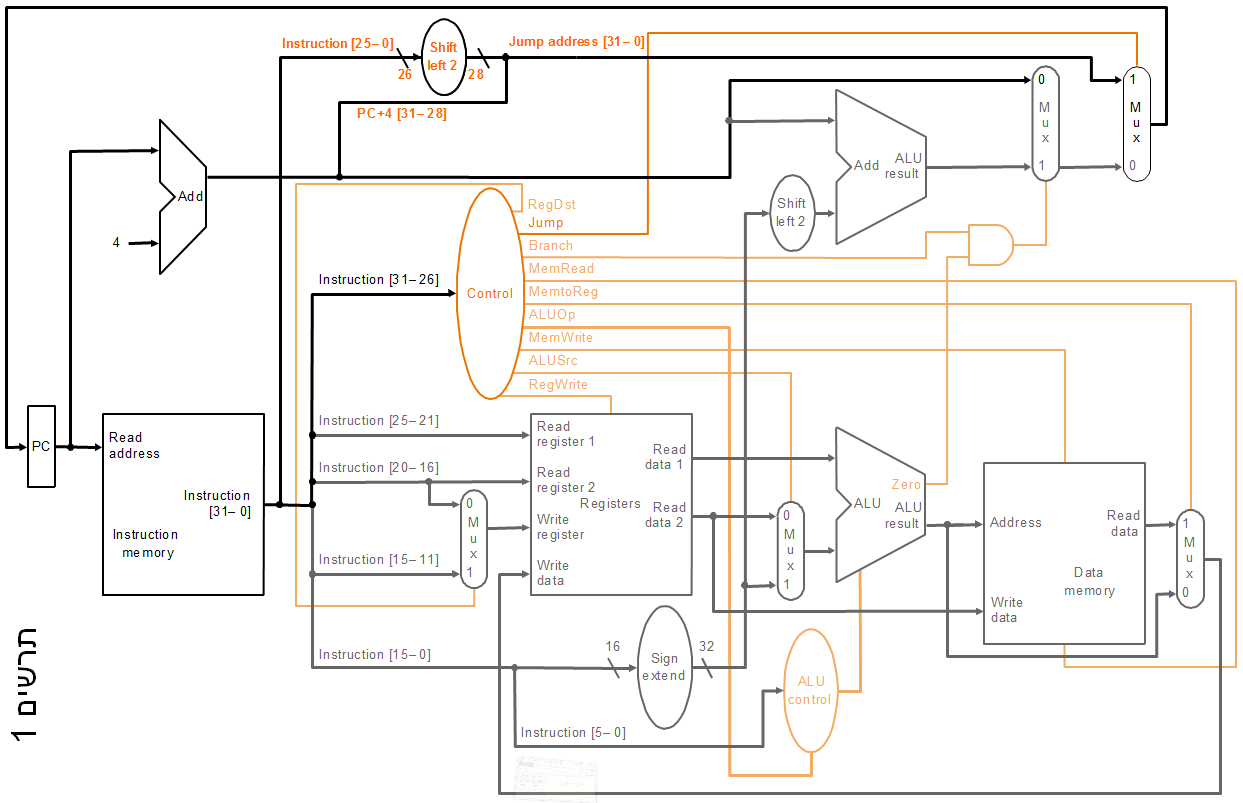


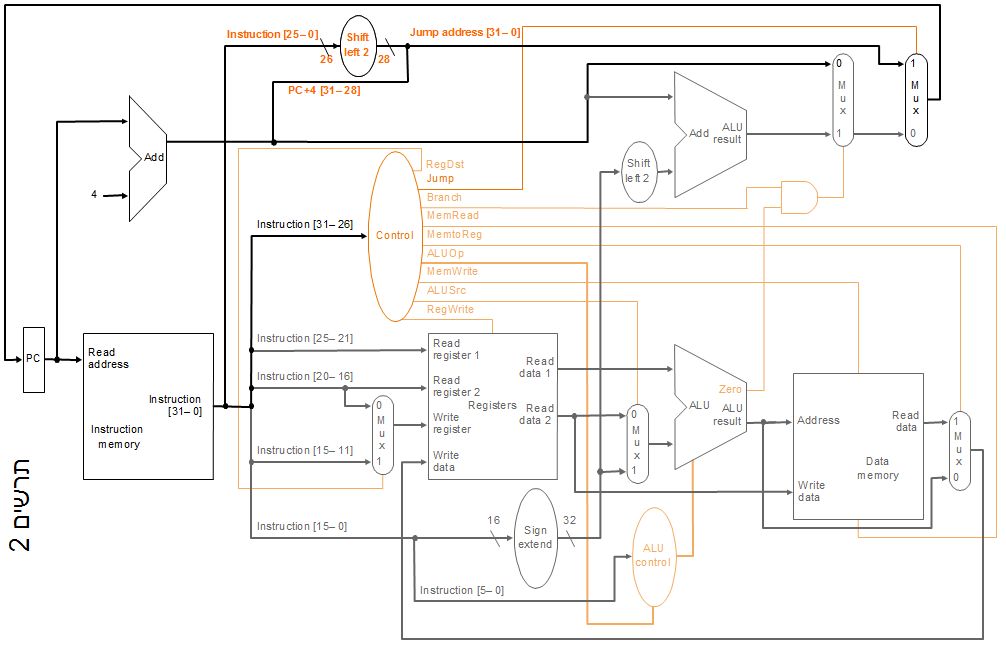
**תשובה**: נדרשת הרחבת ה MUX הנשלט על ידי אות הבקרה RegDst לכניסה שלישית של הערך הקבוע31, דבר המחייב שתי סיביות באות הבקרה. כמו כן, מאחר ולאוגר $ra ($31) נדרש להעביר את כתובת החזרה מהסברוטינה, יש להוסיף ל MUX הנשלט ע"י MemtoReg כניסה נןספת של PC+4 . אותות הבקרה של שני ה MUX הנ"ל הופכים לשתי סיביות.

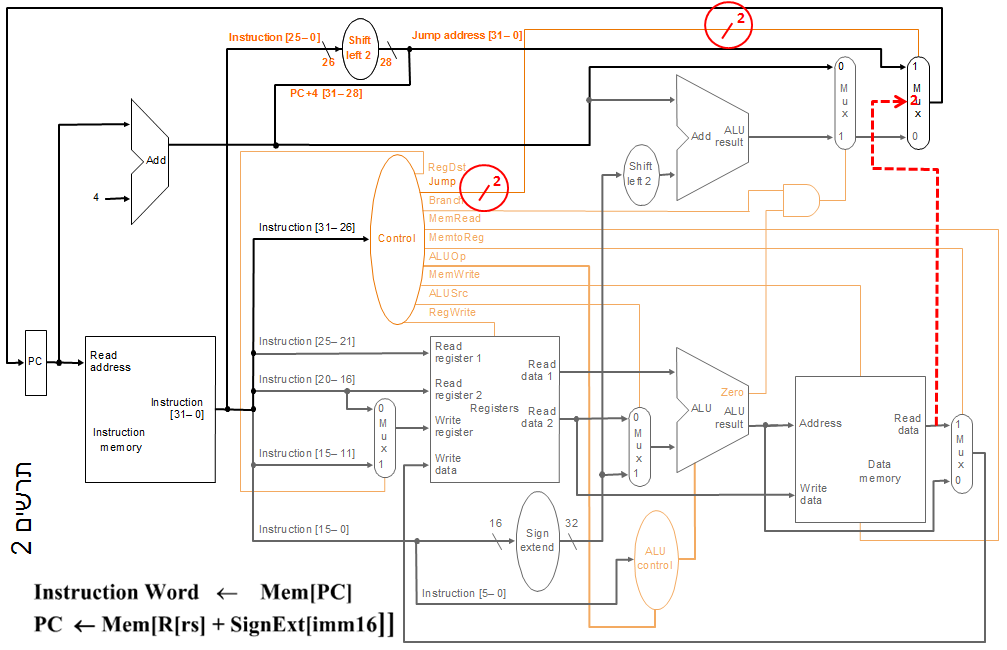


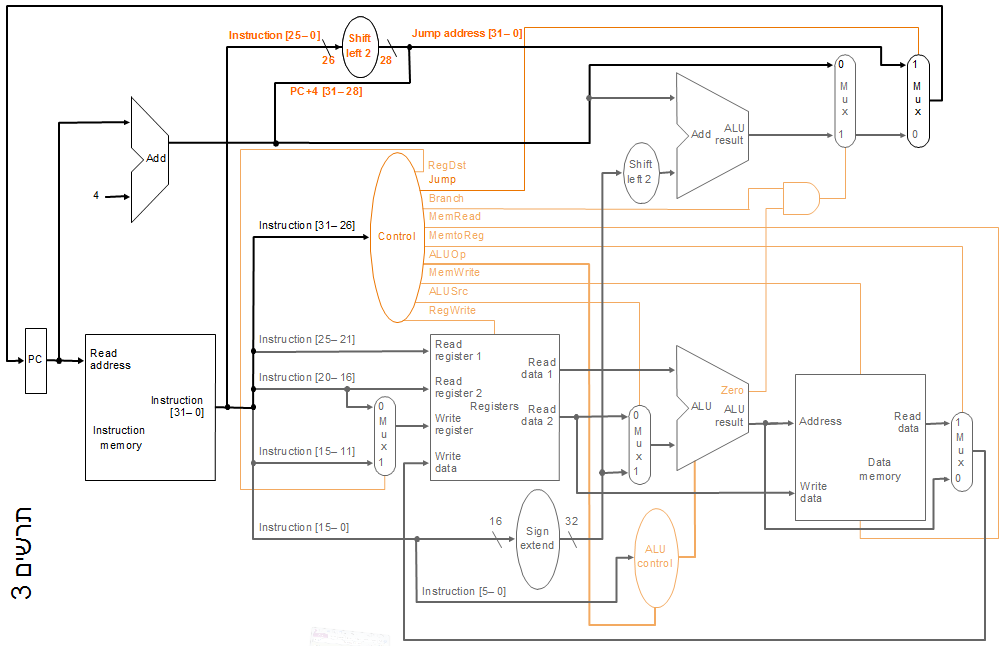
1. (3 נק') פקודת **return** גורמת לחזרה מהסברוטינה. הסבירו האם נדרשת תוספת חומרה למעבד, ובמדה ונדרש, מה התוספת (אין צורך לצייר)?

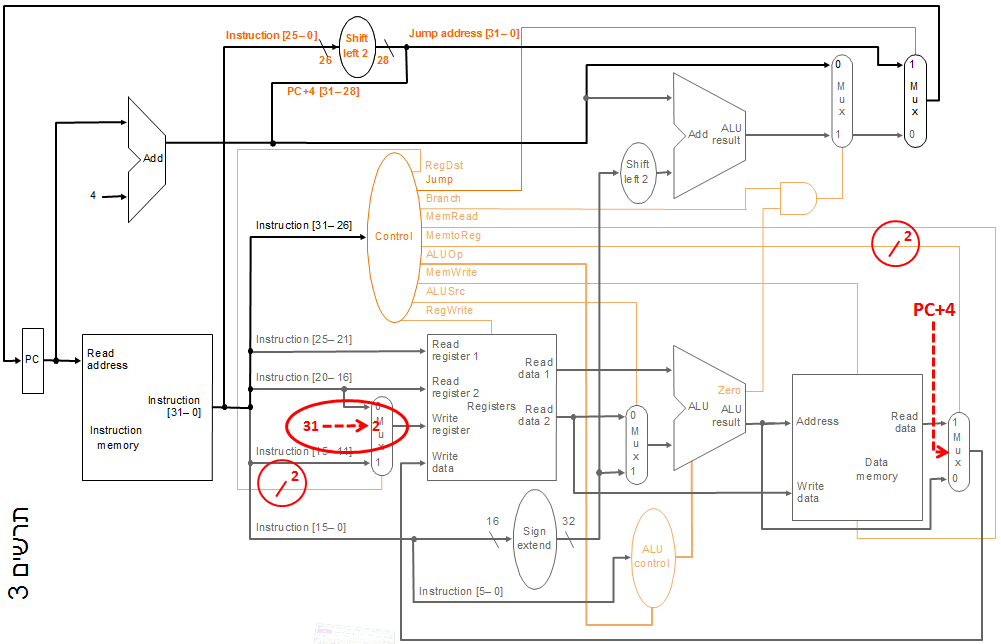
**תשובה**: מאחר ונדרשת מהסברוטינה חזרה לכתובת שנשמרה באוגר $ra ($31), יש ראשית לקרוא את האוגר הזה. יש אם כך להעביר את הערך 31 לאחת מכניסות הקריאה של ה register file, ואת היציאה המתאימה לנתב ל PC. הדבר יחייב MUX חדש באחת מכניסות הקריאה של ה register file, ורחבת ה MUX השולט על ה PC.











**שאלה ב (45 נקודות)**

בשאלה זו אתם נדרשים לממש את הפקודה SWAP המחליפה בין ערכי שני רגיסטרים הנמצאים בקובץ הרגיסטרים. המערכת תמומש במעבד PIPELINE MIPS הנמצאת בתרשים XXX.

1. (5 נקודות) ללא הפקודה החדשה, בקונפיגורציה הזאת, מהם ההשוואות שנעשות ביחידת הFORWARDING בשלב הDECODE? רשמו **במדויק** ופרטו **בקצרה** למה נצרך כל השוואה. האם יש השוואה לצורך פתרון בעיית LOAD HAZARD?

* Detect hazard conditions:
  + - 1a) ID/EX.Rd = IF/ID.Rs
    - 1b) ID/EX.Rd = IF/ID.Rt
    - 2a)EX / MEM.Rd = ID/EX.Rs
    - 2b) EX/ MEM.Rd = ID/EX.Rt
  + (also check for RegWrite, zero, and newest)
* Use these checks to control the forwarding multiplexers

LOADHAZARD נפתר ע"י מערכת אחרת.

1. (10 נקודות) אתם נדרשים לממש את הפקודה ע"י שינויים מינמליים. מותר להגדיל בוררים (mux) קיימים, ולהוסיף בורר יחיד נוסף, לספק כניסות קבועות נוספות, וכן שינויים מינימליים בבקר (הוצאה והכנסה של קוי בקרה וזיהוי opcode חדש). כמובן שבהקשר הנל מותר גם כן להוסיף סיביות לאותות הבקרה שיאגרו באוגרי הצינור. ממשו. בחלק זה לא להתייחס ליחידות הFORWARDING השונים והHAZARD UNIT.

נבצע זאת בשני שלבים. הפקודה החדשה SWAP תהיה בעלת OPCODE מסוים חדש. כמו כן, הקומפיילר אוטומטית תכניס את הכתובת של הרגיסטר השני לתוך כתובת רגיסטר היעד. בשלב הראשון של הפקודה הפקודה תקרא את שני הרגיסטרים ותמשיך הלאה כפקודה רגילה. כאשר הבקר יגלה שמדובר בפקודת SWAP (שלב הDECODE) הוא יבצע העתקה של כל המידע שברגיסטר IF/ID מלבד הOPCODE לתוך עצמו ע"י הבורר החדש. כמובן שדם לא יתן לPC להיכתב מחדש. לתוך סיביות הOPCODE של הרגיסטר הנל ייכנסו OPCODE חדש SWAP2. עקב כך, במחזור שעון הבא הכל יקרה שוב אלא שהפעם המערכת תדע כי מדובר בחלק השני של פקודת הSWAP. במחזור שעון זה, החלק הראשון של הפקודה נמצאת בשלב הEX. כאן ייסכם הרגיסטר עם רגיסטר האפס וימשיך במורד הPIPE כפקודת RTYPE רגילה. הדבר נעשה ע"י שליטה בבוררי הכניסה לALU ע"י סיבית בקרה נוספת (כל המצבים 5והלאה של הבורר יעבירו את ה0 או לחילופין לוגיקת בורר מורכבת יותר של אם הסיבית החדשה דלוקה ה0 עובר).

עבור השלב השני של פקודת הSWAP נדרש להעביר גם כן את יעד הכתיבה של הרגיסטר השני שהוא כתובת הרגיסטר הראשון. כתובת זו מועברת בPIPE גם ככה לצורך מערכת הFORWARDING של שלב הEX. ניקח אותה משם ונכניסה גם כן לבורר היעד שיש בשלב הEX ונרחיב את קו הבקרה שלו ל2 סיביות.

בזה תמו השינויים מלבד עדכון מערכת הFORWARDING של שלב הEX.

1. (10 נקודות) כיצד תשתנה מערכות הFORWARDING השונים (לציין את שניהם!)?

ראשית נשים לב כי יחידת הFORWARDING של שלב הEX חייבת להתעדכן שכן אם בשלב SWAP1 אנו מתעתדים לכתוב לתוך הרגיסטר השני הרי שכפי שהמערכת כרגע, יחידה זו תכניס את הערך החדש לALU בזמן SWAP2. ולכן צריך שיחידת הFORWARDING תעשה את עבודתה נאמנה בכל שאר הפקודות מלבד פקודת SWAP בחלק השני! בחלק זה היא תבטל את הבדיקה משלב הMEM בלבד!

כלומר קו הבקרה תבדוק אם מדובר בSWAP2 לא לאפשר FORWARD משלב הMEM.

מכאן התשובה איננה נכונה!- **כנל ליחידת הFORWARDING שבלשב הID. גם הוא לא אמור להתעדכן משלב הEX במקרה הזה. כלומר קו הבקרה לא תאפשר קידום משלב הEX אם SWAP2 הוא המצב כרגע בID**. – FORWARDING הזה מיועד רק לפקודות BRANCH ואינו רלוונטי. הוא אינו מחליף ערכים רק בוחן השוואה

1. (5 נקודות) כיצד תשתנה הHAZARD DETECTION UNIT ? האם המערכת עדיין תצליח לפתור את כל בעיות וסוגי הHAZARDS שלמדתם בכיתה? במידה ויש השוואות נוספות שצריכים להעשות ציינו זאת. במידה ויש ייעול ציינו זאת.

DATA HAZARDS אינם קשורים למערכת זו וכבר נפתרו.

LOAD HAZARDS נפתרים בצורה הרגילה (BUBBLE INSERTION) אך ניתנים לייעול שכן רק רגיסטר אחד באמת צריך להיבדק בשלב של SWAP1 שכן הרגיסטר השני ייקרא שוב בSWAP2 כאשר הערך כבר יעודכן.

אין שום שינוי למערכת מלמבד הערה זו.

1. (5 נקודות) מה המחיר בזמן ההרצה של תוכנית כלשהי אם 5% מסך הפקודות בתוכנית (התוכנית מבצעת הרבה מאוד פעולות של SORT) הם מסוג זה? האם המחיר הנ"ל תלוי בHAZARDS חדשים מתוקף הפקודה החדשה? כיצד?

אין שום תלות בHAZARDS חדשים. נוסיף מחזור יחיד בלבד לפקודות הSWAP על פני שאר הפקודות ולכן אם 5% מהזמן יש פקודה מסוג זה הרי שזמן ההרצה מתארך כך ש

תחת הזנחת מילוי הצינור.

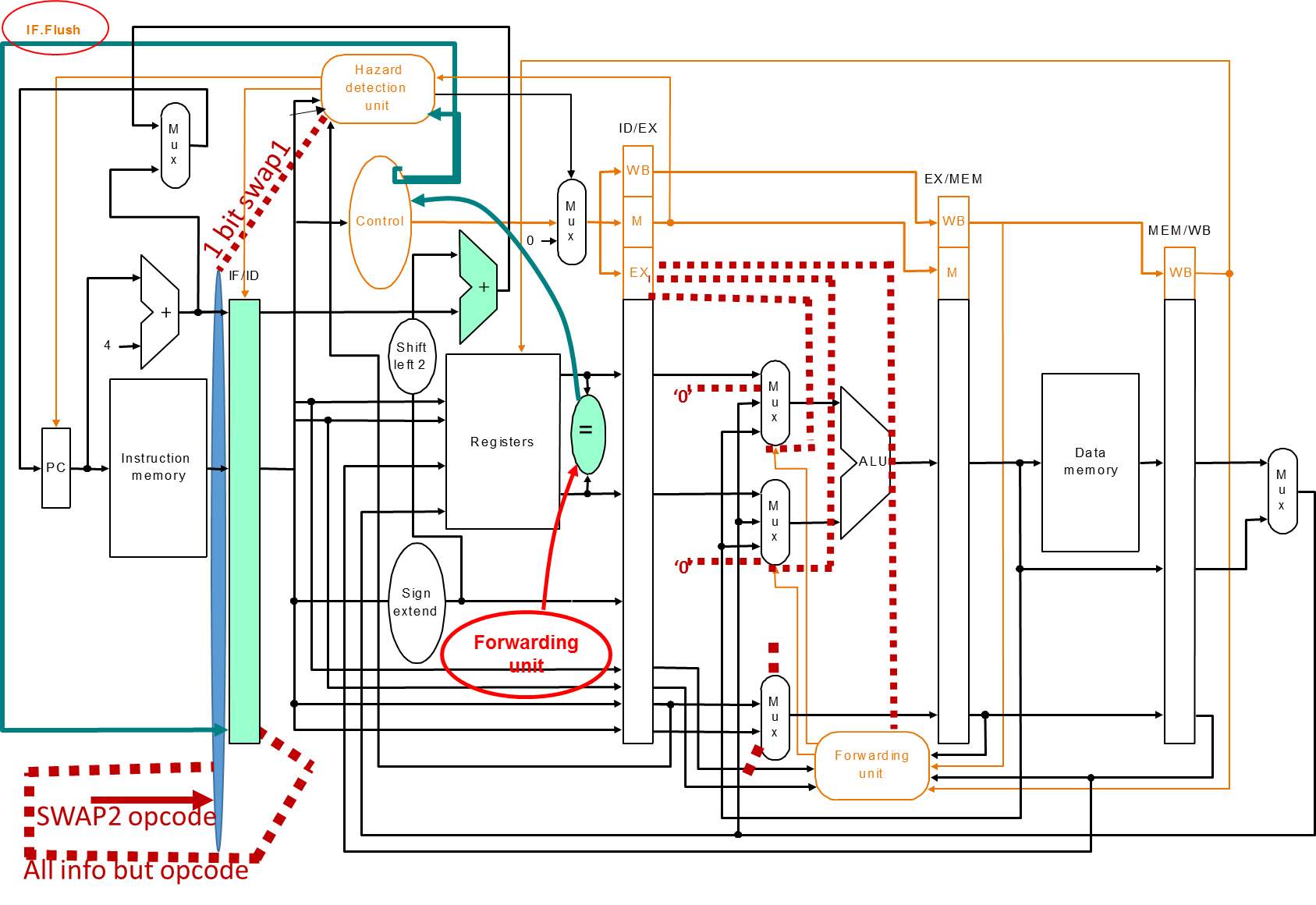
1. (5 נקודות) האם ניתן לממש את הפקודה הנ"ל בין שני ערכים האגורים בזכרון ולא בקובץ הרגיסטרים? כיצד? ענו תחת הגבלות מעין אלו לעיל (ללא שינויים במערכים קיימים למעט קוי בקרה ובוררים).

ציינו מהם הבעיות העקרוניות וכיצד חייבים לפתור אותם. בכל מקרה אין לחרוג ממחיר של 2 מחזורים לפקודת הSWAP .

ללא שינויים לא ניתן לעמוד בתנאים. לא ניתן לבצע קריאה, כתיבה, קריאה , כתיבה בשום אופן מהזכרון ב2 מחזורים. ולכן גם לא ניתן לקרוא ולכתוב לשני מקומות אחרים בזכרון, קל וחומר להחלפה בין שניים.

1. (5 נקודות) האם ניתן לבצע זאת ע"י שינויים של מערכת הזכרון? למרות השינויים האם ניתן לבצע זאת במחזור יחיד? באיזה מידה זה שונה מאשר ביצוע שינויים מעין אלו לצורך ביצוע הפקודה בקובץ הרגיסטרים (רמז, תזמון השעון).

עם שינויים הכל כבר אפשרי. הוספת פורט קריאה נוסף, העברה של יותר מיעד קריאה אחד וכן כתיבה מתוזמנת. הבעיה היא הכתיבה שאיננה מתוזמנת שעון. נצטרך לוודא שתזמון הקריאה והכתיבה נעשה בצורה שווה במקביל ושלא יתבצע SKEW בין שני חלקי הפקודה באותו מחזור שעון כך שייווצר מרוץ. צריך ליצור חוצץ בין מערכת הכתיבה לזכרון לבין מערכת הקריאה שיתוזמן כך שלא ייווצר מצב בשום פקודה של קריאה וכתיבה מהזכרון באותו זמן כך שמידע שנקרא במחזור אחד לא יכתב למקום אחר במחזור הבא (יצירת LOOP ללא חוצץ הוא בעייתי).



**שאלה ג (30 נקודות)**

שאלת תכנון ומימוש בשפת תכן Verilog:

תיאור כללי של הבעיה:

* נדרש: רכיב המממש מיון-בועות "bubble sort" הממומש לפי האלגוריתם שנידון בהרצאה.
* הרכיב מקבל מטריצה דו מימדית בגודל 8 על 32 בשם in (כלומר שמונה מלים בנות 32 ביט כל אחת). הרכיב ממיין את המלים לפי הסדר ומוציא מטריצה זהה בגודלה ומסודרת out .
* חובה להשתמש במבנה generate ולהשתמש בהפניות היררכיות בקוד.
* חובה להגדיר את מספר המלים וגודלן כפרמטר.

סעיפים:

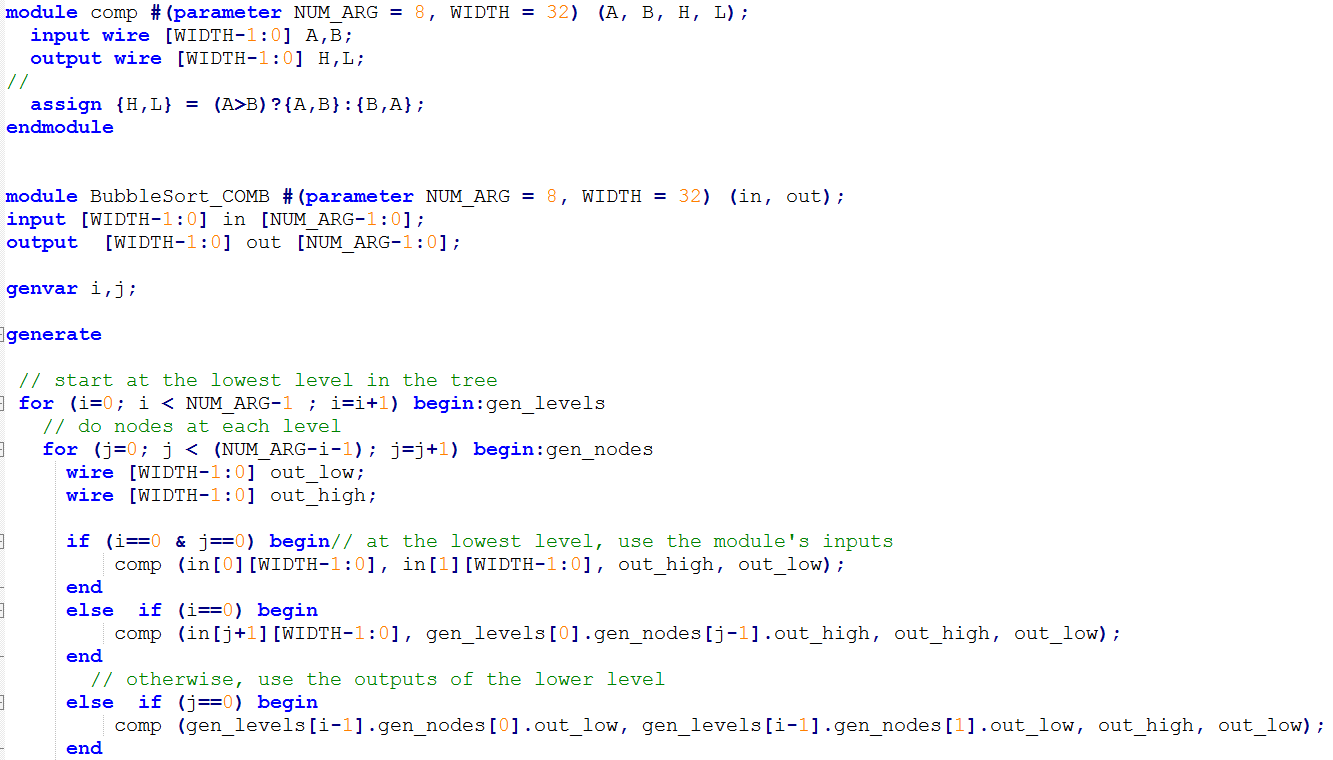
* 1. (25 נק) תכנן את המעגל צירופית לחלוטין.
  2. (5 נק) תאר בנקודות אילו שינויים היית עושה בכדי להפוך את הקוד לסינכרוני (עם שעון clk) ובמבנה pipeline. אין צורך לממש אך מספיק לרשום בנקודות עיקריות מה השינויים המרכזיים הדרושים.

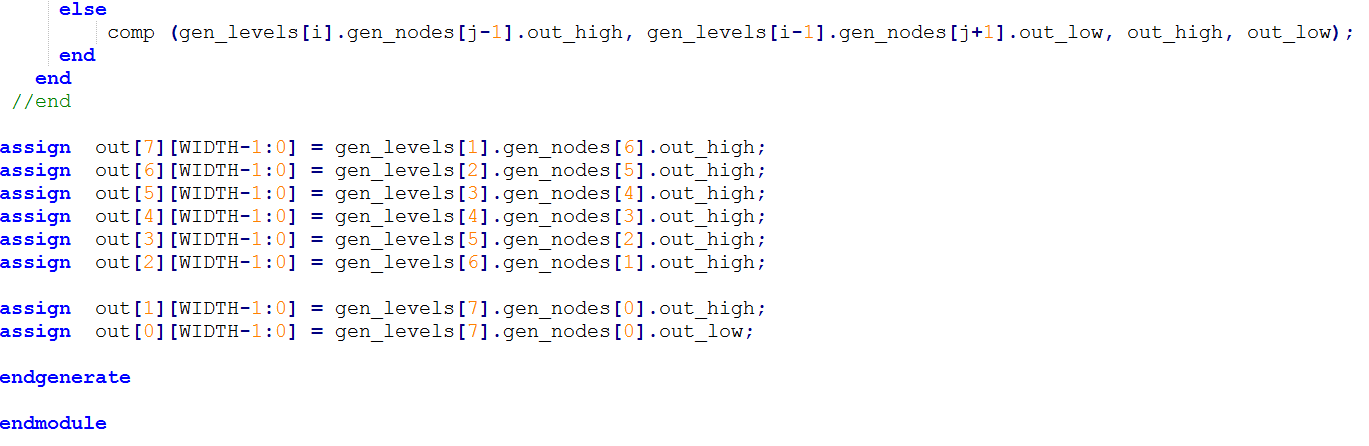
**דגש: ניקוד משמעותי ירד לכתב לא ברור\ לא מסודר\ מקושקש מדיי - תעבדו עם עפרון או במחברת טיוטא ורק קוד סופי למסגר – רק מה שימוסגר ייבדק.**

**פתרון-** זהו מקום לקוד סופי (טיוטא במחברת):

\_\_\_מודול\_ממיין\_בועות:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

פתרון:





סעיף ב:

