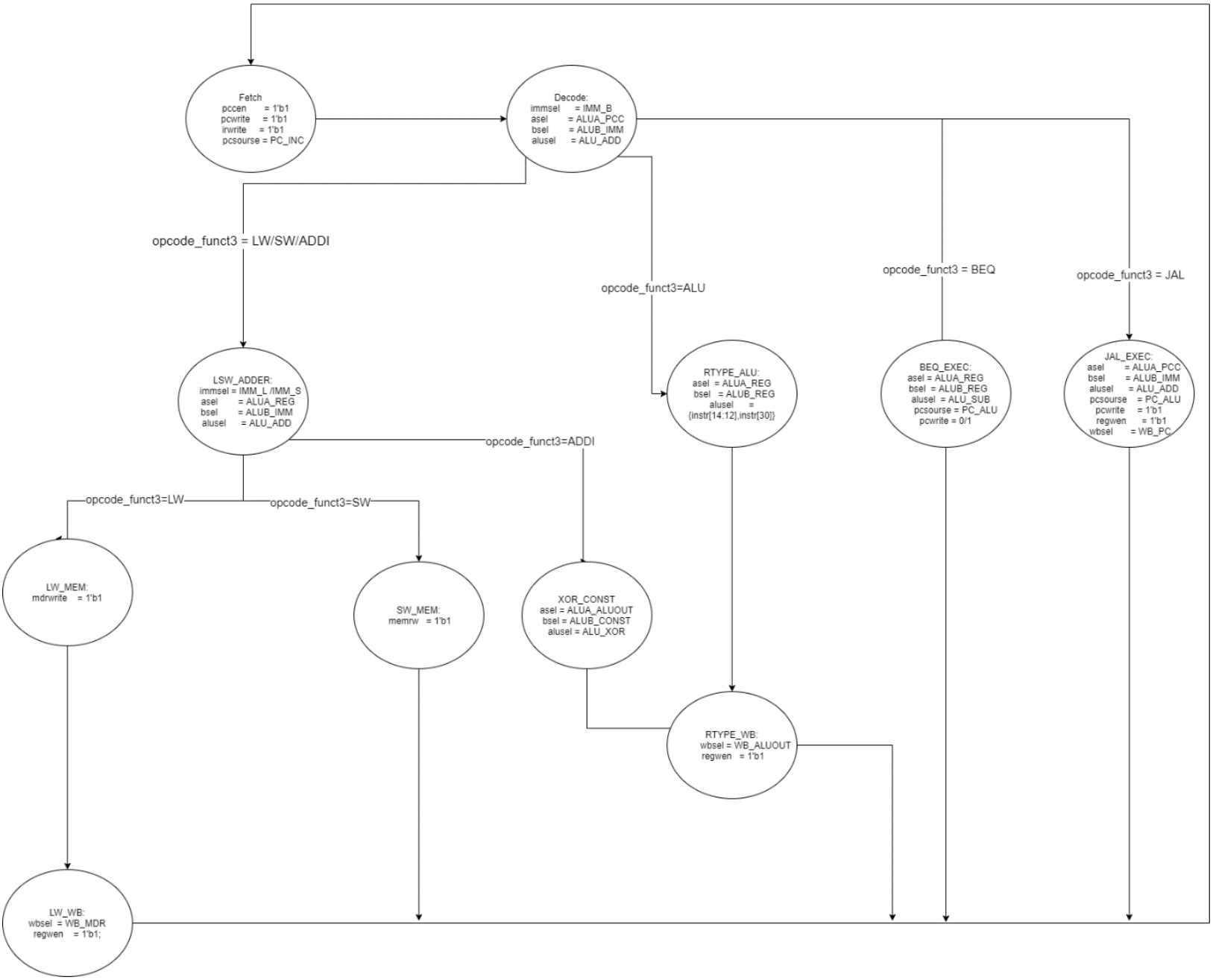
אופיר אלדר	314722950
בן הלפרין	209086578

סימולציה 3 חלק יבש

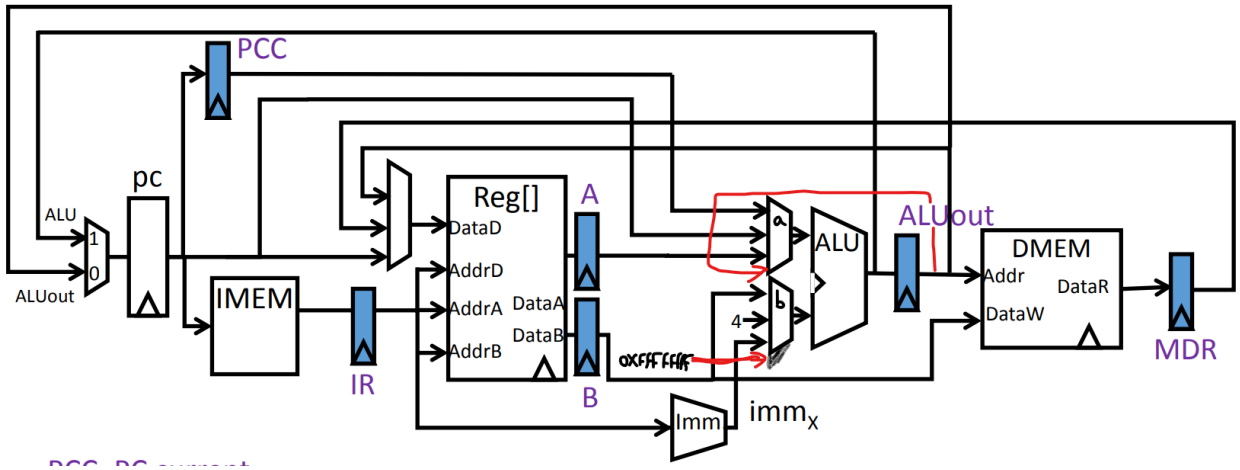
:2.1

שרטוט דיאגרמת מצבים:



כפי שניתן לראות, הוספת התמיכה בפקודה החדשה דרשה שינוי מינימלי במכונת המצבים, כאשר המצב היחיד שנוסף הוא XOR_CONST. מהלך ביצוע הפקודה הוא כזה: לאחר שלבי ה fetch וה decode הזהים לכל סוג פקודה, נעבור למצב LSW_ADDER. עד עכשיו מצב זה שימש לפקודות lw ו sw בלבד. תפקידו הוא לחשב כתובת יעד באמצעות חיבור בין כתובת הבסיס השמורה ברגיסטר rs1 לבין immediate שמתאר הזחה לעומת כתובת הבסיס. זוהי בדיוק הפעולה הראשונה שמבוצעת ע"י addi: חיבור בין תוכן rs1 לבין immediate. לכן פקודת addi תשתמש במצב הזה. נשים לב כי נדרש שהמצב יתמוך בקריאת ה immediate בפורמט בו הוא כלול בפקודה addi, אבל פורמט זה זהה לפורמט בו הוא כלול בפקודה lw, ולכן נקבע immsel = IMM_L. לאחר מכן נדרש לבצע פעולת XOR בין תוצאת החיבור לבין קבוע. פעולה זו לא התרחשה עד כה באף אחת מהפקודות הקיימות ולכן דרשה יצירת מצב חדש, שנקרא XOR_CONST. מצב זה דורש מהALU לבצע בהכרח פעולת XOR, בין הרגיסטר ALUOUT בו נמצאת כרגע פעולת החיבור מהמצב הקודם, לבין הקבוע 0xffffffff שהתווסף ככניסה לmuxb כפי שנראה בשרטוט המעבד. השלב האחרון שנדרש בביצוע הפקודה הוא כתיבת תוצאת הXOR לרגיסטר rd, אבל זו בדיוק הפעולה שמתבצעת במצב RTYPE_WB ששימושו המקורי היה כתיבת תוצאות פעולות R-TYPE לתוך rd. לכן נעבור מתוך XOR_CONST ל RTYPE_WB ומשם חזרה ל Fetch, מה שמציין את סיום ביצוע הפקודה.

שרטוט מעבד:

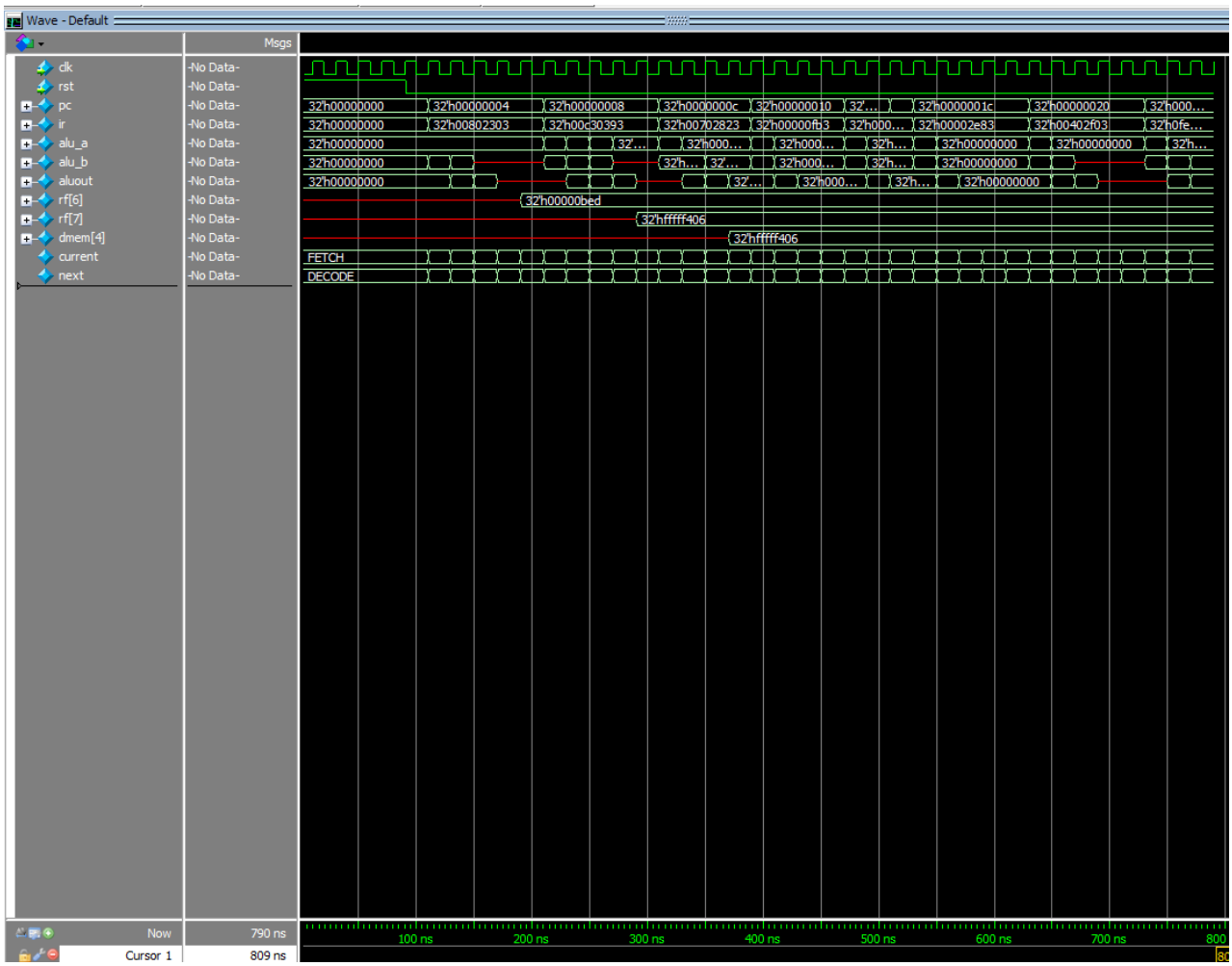


התוספות שלנו מסומנות באדום.

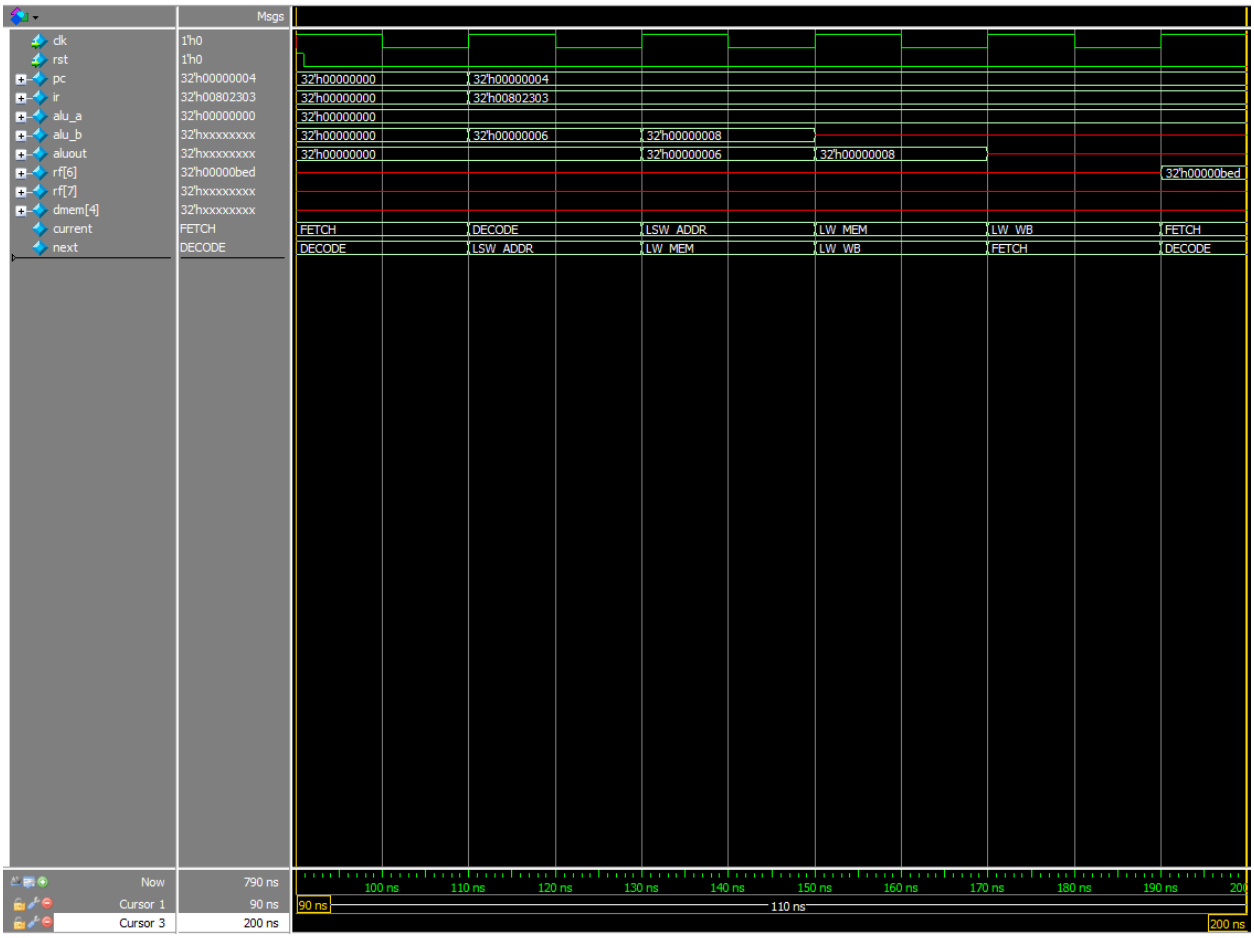
הוספנו לmuxa חיבור לתוצאת הalu על מנת לתמוך בשימוש בתוצאת החיבור בפעולת ה xor שבאה לאחר מכן. (ולכן לסלקטור של הmux הזה התווסף ביט נוסף).

הוספנו לmuxb חיבור לקבוע 0xffffffff, על מנת ביצוע פעולת הxor איתו. (ולכן גם לסלקטור של הmux הזה התווסף ביט נוסף).

:2.3

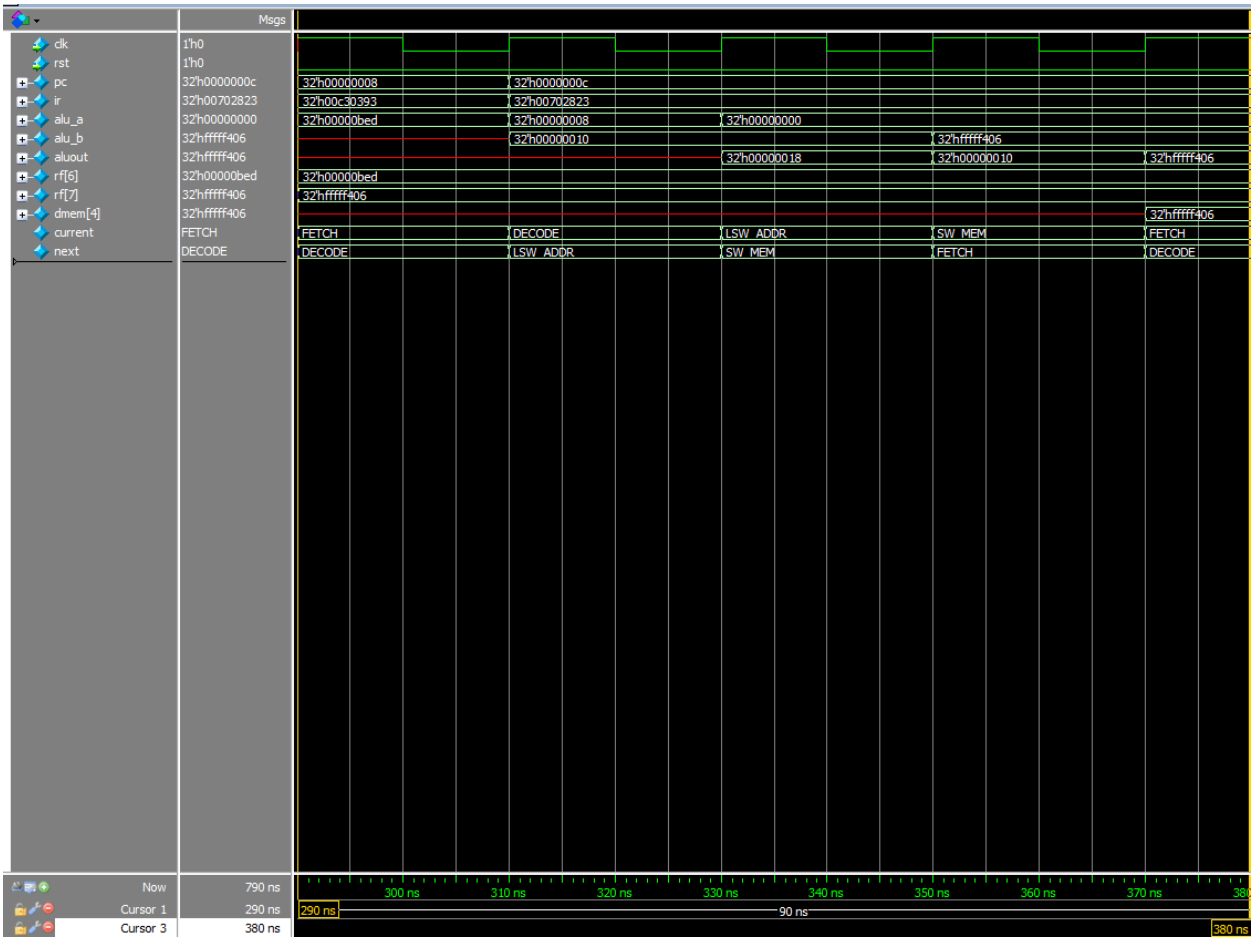


מבט כללי על כל מהלך הסימולציה.

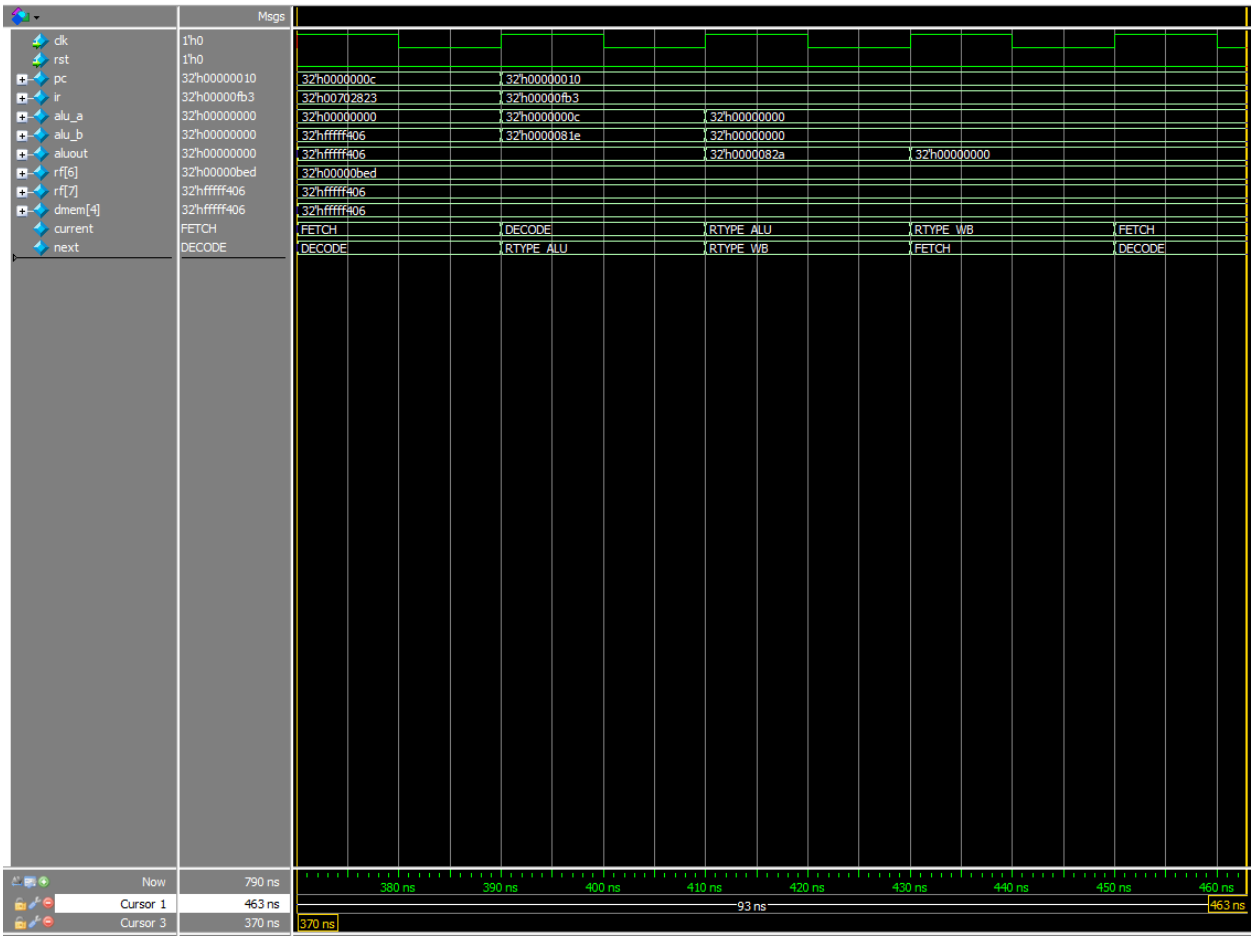


הפקודה הראשונה בtest היא פקודת lw שטוענת את התוכן של הכתובת 8 בזיכרון לרגיסטר t1, שהוא הרגיסטר באינדקס 6. ניתן לראות כי מצבי המכונה במהלך ביצוע הפקודה תואמים לחמשת המצבים של LW בדיאגרמת מכונת המצבים. בנוסף, לאחר שלב LSW_ADDER הכתובת ב aluout ממנה ייטען הזיכרון היא אכן 8, ולאחר שלב LW_WB האחרון, ב[6]rf שהוא t1 מופיע הערך הדרוש.

הפקודה השנייה היא פקודת addi שהוספנו. מצבי המכונה תואמים את חמשת המצבים של addi בדיאגרמת מכונת המצבים, כולל שימוש במצב XOR_CONST החדש. הפקודה מוסיפה לערך בt1 את הקבוע 12: ניתן לראות שבשלב LSW_ADDR בו מתבצע החיבור, הערך של alu_a הוא הערך שנטען קודם לתוך t1, והערך של alu_b הוא 12. הערך של aluout מתעדכן לתוצאת החיבור הנכונה בשלב הבא, שהוא שלב ביצוע הxor. כעת alu_a מתעדכן להיות תוצאת החיבור שהייתה בaluout, בעוד שalu_b מתעדכן להיות הערך הקבוע 0xffffffff. בשלב הכתיבה לרגיסטר כבר ניתן לראות את תוצאת הפעולה בaluout, ולאחר עליית שעון ניתן לראות שהיא נכתבה לrf[7], כלומר לt2.

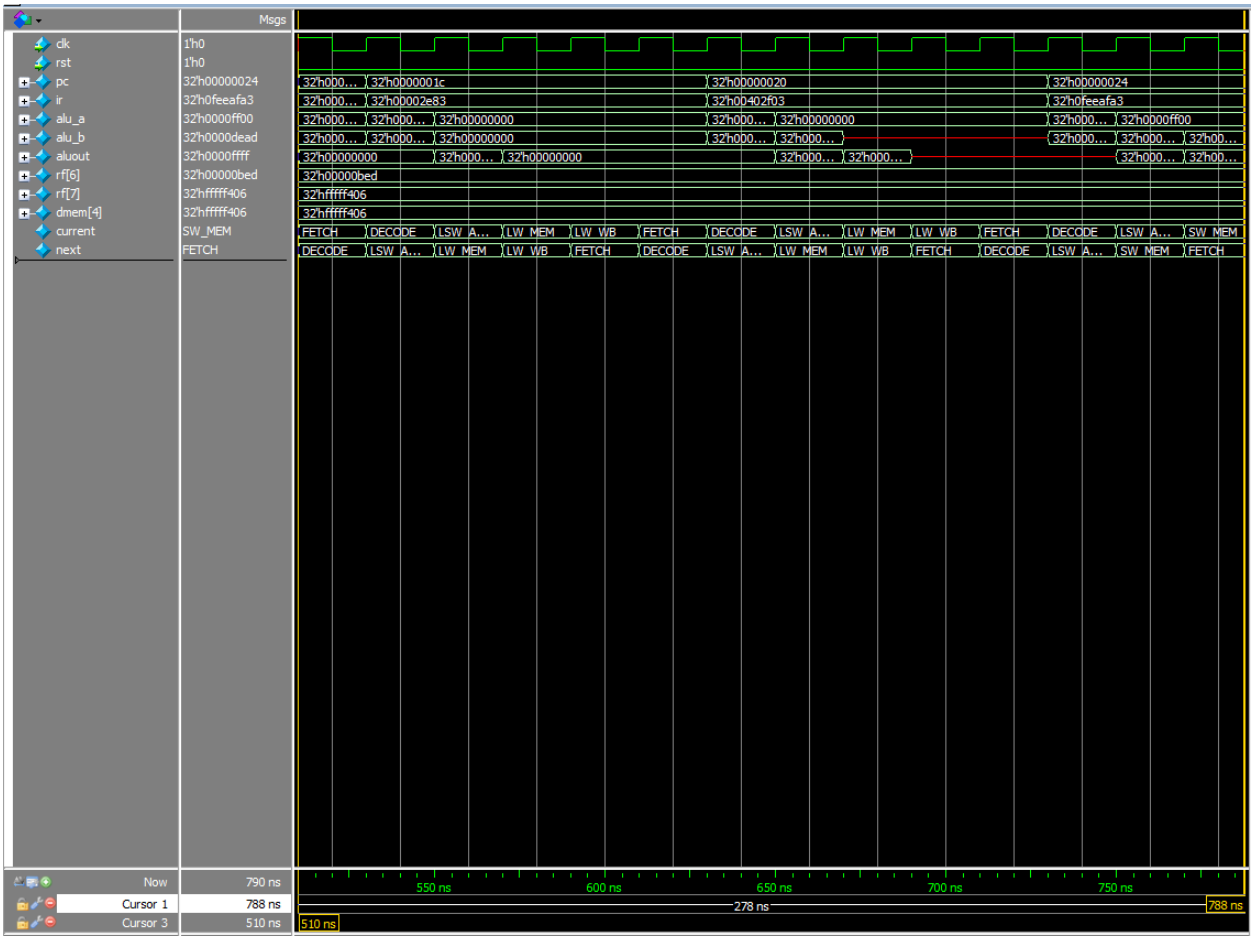


הפקודה הבאה היא פקודת sw ששומרת את הערך שנמצא בt2 בכתובת 16 בזיכרון. פקודת sw קצרה יותר מהקודמות ועוברת דרך 4 מצבים, כלומר אורכת 4 מחזורי שעון. לאחר שלב LSW_ADDR שמחשב את הכתובת בזיכרון בה יישמר הערך ברגיסטר, ניתן לראות כי הערך השמור בaluout הוא 10 בבסיס הקסדצימלי, או 16 בבסיס דצימלי כנדרש. לאחר שלב SW_MEM בו מתבצעת הכתיבה לזיכרון, הערך בזיכרון נהיה התוצאה שהייתה שמורה בt2.



לאחר מכן מתבצעת פקודת add שמכניסה לרגיסטר t6 את הערך 0. בשלב RTYPE_ALU בו מתבצע החיבור ניתן לראות כי גם alu_a וגם alu_b מקבלים את הערך 0, וזה גם ערך תוצאת החיבור בaluout לאחר עליית שעון.

הפקודה הבאה מבצעת השוואה בין t6 ל0 – בפקודה הקודמת קבענו את ערכו של t6 ל0 ולכן תתקיים קפיצה לlabel finish שמציין את הכתובת 18 שחושבה כבר בשלב הDECODE. לאחר סיום BEQ_EXEC הכתובת בpc מתעדכנת להיות הכתובת שחושבה.



סוף הסימולציה מבצע 3 פעולות lw,lw,sw. מטרתן לטעון לt4 את הערך 0 ששמור בx0, לטעון לt5 את הערך ששמור בכתובת 4, ולשמור את הערך שנטען לt5 בכתובת FF בבסיס הקסדצימלי שהיא 255 בבסיס דצימלי.
לאחר מכן התוכנית נכנסת ללולאה אינסופית כיוון שהיא ממשיכה לבדוק האם t6 מכיל את הערך 0 בלי לשנות אותו במהלך הלולאה, אבל הסימולציה מפסיקה באופן אוטומטי.
הפקודה lw כוללת 5 מצבים בעוד ש sw כוללת 4 מצבים, כלומר בסך הכל שלב זה של התכנית כולל 14 מצבים כלומר אורך 14 מחזורי שעון, כפי שניתן לראות בסימולציה.