**סימולציה 3**

מוגש ע"י:

|  |  |
| --- | --- |
| 315817155 | מייק אלכסנדרובסקי |
| 318288925 | דניאל טנסיינקו |

2.4 –

בפקודת ‘addi’ ישנו שימוש ב-ALU פעמיים. נצטרך להוסיף למכונת המצבים שלושה מצבים חדשים שיהיו אחראיים על הפקודה.

A picture containing timeline

Description automatically generated

* ITYPE\_ALU – מבצע את החיבור הראשוני בין rs1 ל-immediate ושומר את התוצאה ברגיסטר ALUout.
* ITYPE\_ALU2 – מבצע את פעולת ה-xor בין התוצאה השמורה ב-ALUout לבין המספר 0xffffffff, ושומר את התוצאה ברגיסטר ALUout.
* ITYPE\_WB – מבצע כתיבה חזרה של התוצאה הסופית השמורה ב-ALUout לרגיסטר rd.

Diagram

Description automatically generated

השינויים שבוצעו במעבד:

* חיבור מוצא הרגיסטר ALUout ככניסה לבורר Asel.
* חיבור המספר 0xffffffff ככניסה לבורר Bsel.

A screenshot of a computer

Description automatically generated2.3 –

ניתן לראות שב-91ps, rst יורד ל-0 והמעבד מתחיל לעבוד. בעליית השעון הבאה, הפקודה הראשונה נקראת (שלב ה-fetch) ו-pc מתעדכן ל-pc+4. הפקודה היא ‘lw’.

ב-210ps, הפקודה הסתיימה והמעבד מקבל פקודה חדשה, גם כאן ps מתעדכן ועולה ב-4. ננתח את ir:

0000 0000 1100 0011 0000 0011 0001 0011

func3

rs1 (6, x6=t1)

opcode

rd

Imm (12)

זוהי הפקודה ‘addi t1, t1, 12’.

נחשב את התוצאה ונוודא שהיא תקינה:

t1 = 0xfffff453 (-2989)

t1 = t1 + 12 = 0xfffff45f (-2977)

t1 = t1 ^ 0xffffffff = 0x00000ba0 (-1120)

בהמשך מתקבלות הפקודות המתאימות ל-imem. לאחר ביצוע פקודת ה-‘addi’ תכולת הרגיסטר t1 נרשמת בזיכרון (dmem) בכתובת 16:

Graphical user interface, text, application

Description automatically generated

כפי שניתן לראות, הכתובת 16 מכילה את המספר שחושב.