תרגול 9

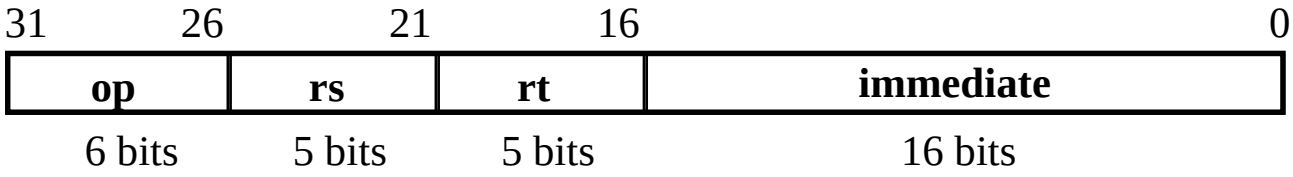
Single Cycle Architecture

מימוש פקודות - התהליך

- ראשית, מתארים מה הפקודה עושה בשלבים:
  - מעבר נתונים (מקורות ויעד) תוך שימת דגש על האופרנדים – זיכרון/רגיסטרים
  - פעולות אריתמטיות שנעשות
- שנית, מממשים
  - מימוש מעבר הנתונים במערכת הקיימת:
    - האם הקווים הקיימים מספקים או
    - צריך להוסיף קווים נוספים
  - מימוש הבקרה במערכת הקיימת:
    - קביעת ערכי הבקרה ביחידות הבקרה הקיימות
    - הוספת mux-ים במידת הצורך והגדרת קווי הבקרה שלהם
    - עדכון פונקציות קווי הבקרה הקיימים ע"י שילוב ערכי קווי הבקרה של הפקודה החדשה

דוגמא – הפקודה addi

- addi $t1, $t2, immediate

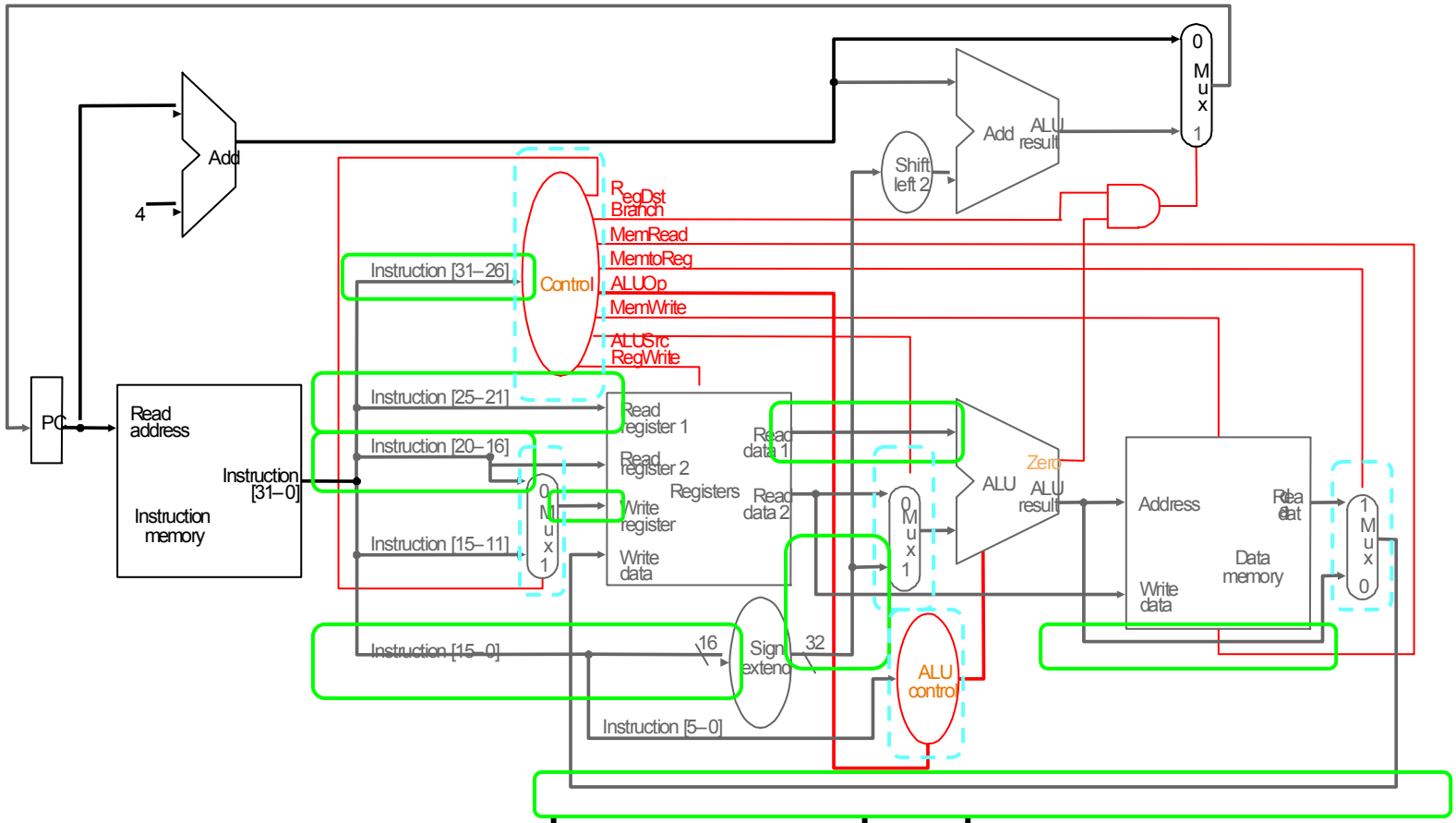


- תיאור:
  - פקודה נלקחת מזיכרון הפקודות ורגיסטר ה-pc מקודם
  - הרגיסטר $t2 (rs) נקרא ממאגר הרגיסטרים (על-פי סיביות 21-25)
  - immediate עובר הרחבה ל-32 סיביות ע"י מריחת הסימן
  - ה-ALU מחשב את סכום הערך שנקרא ממאגר הרגיסטרים ו-immediate מורחב הסימן.
  - תוצאת החישוב נכתבת אל מאגר הרגיסטרים לרגיסטר שמספרו נתון בסיביות 16-20 (rs) .

דמיון רב ל- lw אך ללא הגישה למאגר הזיכרון

addi

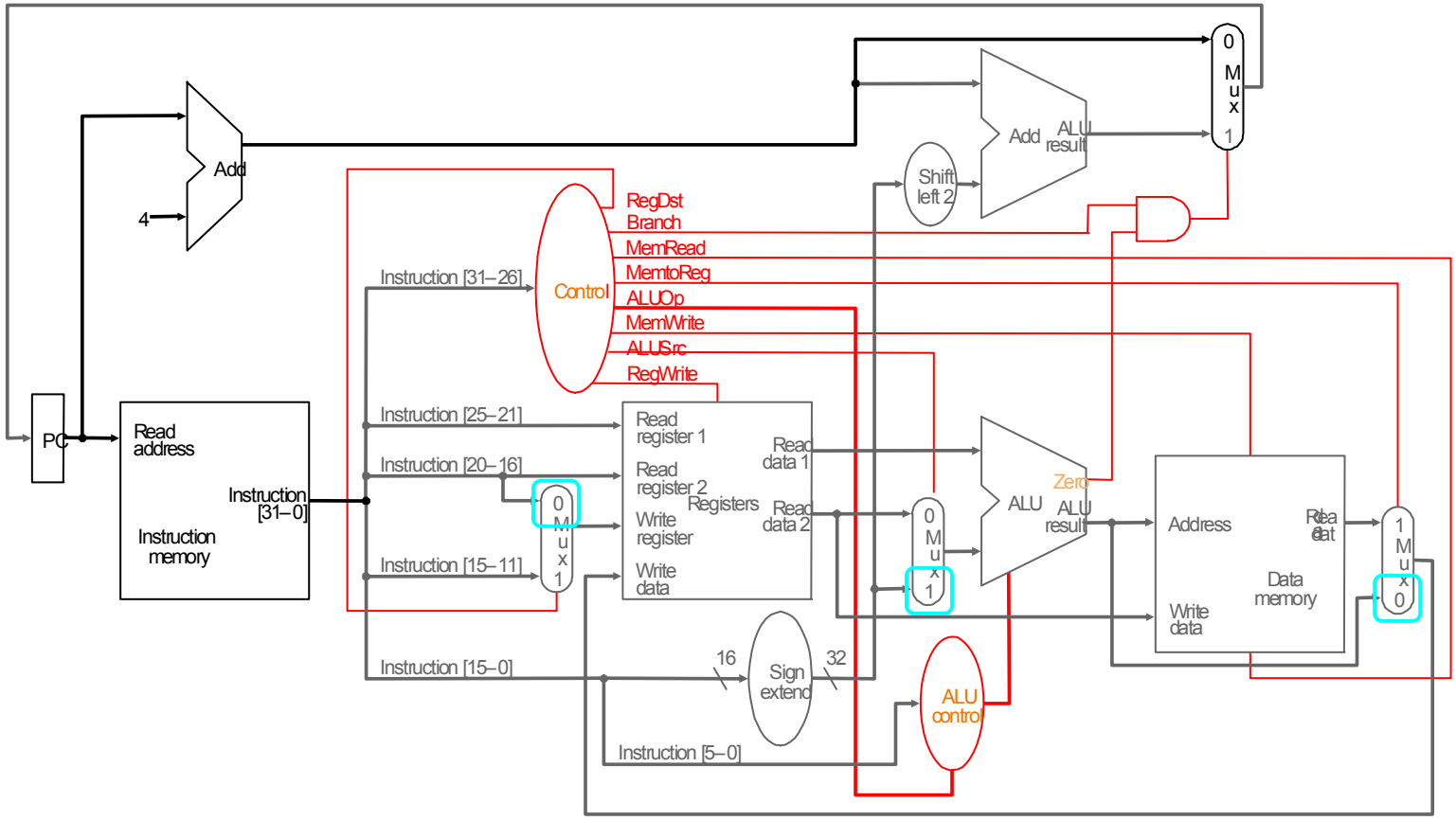
- האם קווי הנתונים הקיימים מספקים ? כן
- האם יחידות הבקרה מספקות ? כן



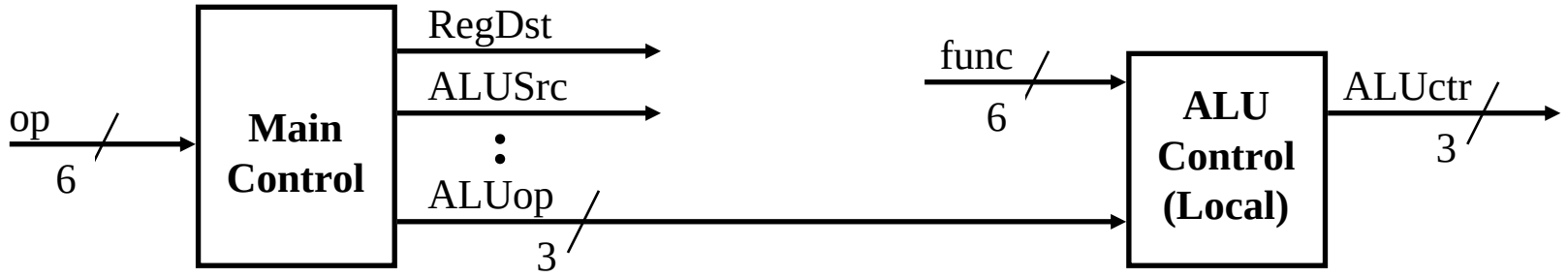
קביעת ערכי הבקרה

Instruction	RegDst	ALUSrc	Memto-Reg	Reg Write	Mem	Mem	Branch	ALUOp1	ALUp0
R-format	1	0	0	1	0	0	0	1	0
`lw`	0	1	1	1	1	0	0	0	0
`addi`	0	1	0	1	x	0	0	0	0

add



עדכון פונקציות קווי הבקרה הקיימים

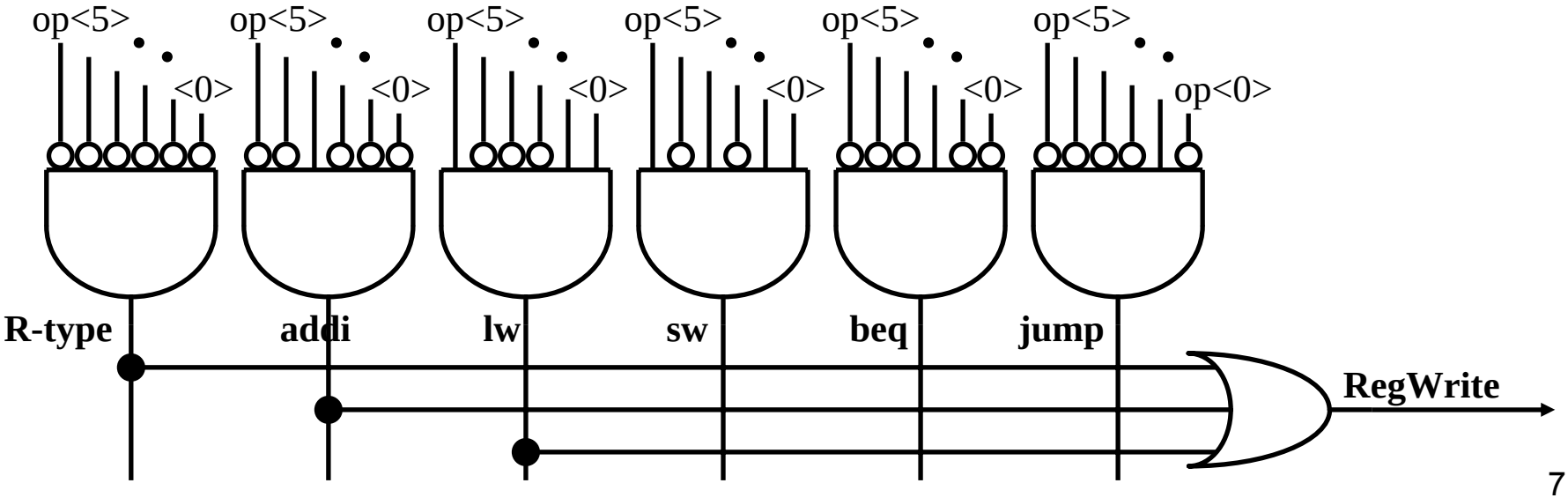


op	00 0000	00 1000	10 0011	10 1011	00 0100	00 0010
	R-type	***addi***	**lw**	**sw**	**beq**	**jump**
RegDst	1	0	0	x	x	x
ALUSrc	0	1	1	1	0	x
MemtoReg	0	0	1	x	x	x
RegWrite	1	1	1	0	0	0
MemWrite	0	0	0	1	0	0
Branch	0	0	0	0	1	0
Jump	0	0	0	0	0	1
ExtOp	x	0	1	1	x	x
ALUop (Symbolic)	"R-type"	Add	Add	Add	Subtract	xxx
ALUop <1>	1	0	0	0	0	x
ALUop <0>	0	0	0	0	1	x

Implementation of *RegWrite*

op	00 0000	00 1000	10 0011	10 1011	00 0100	00 0010
	R-type	**addi**	**lw**	**sw**	**beq**	**jump**
RegWrite	1	1	1	0	0	0

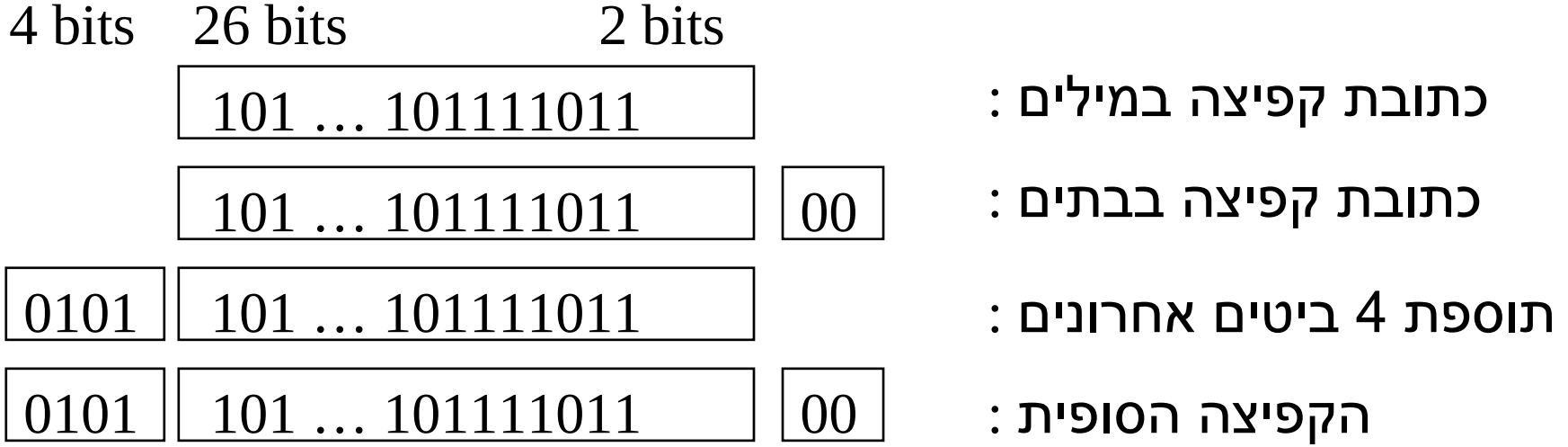
- **RegWrite** = R-type + addi + lw
  = !op<5> & !op<4> & !op<3> & !op<2> & !op<1> & !op<0> (R-type)
  + !op<5> & !op<4> & op<3> & !op<2> & !op<1> & !op<0> (addi)
  + op<5> & !op<4> & !op<3> & !op<2> & op<1> & op<0> (lw)



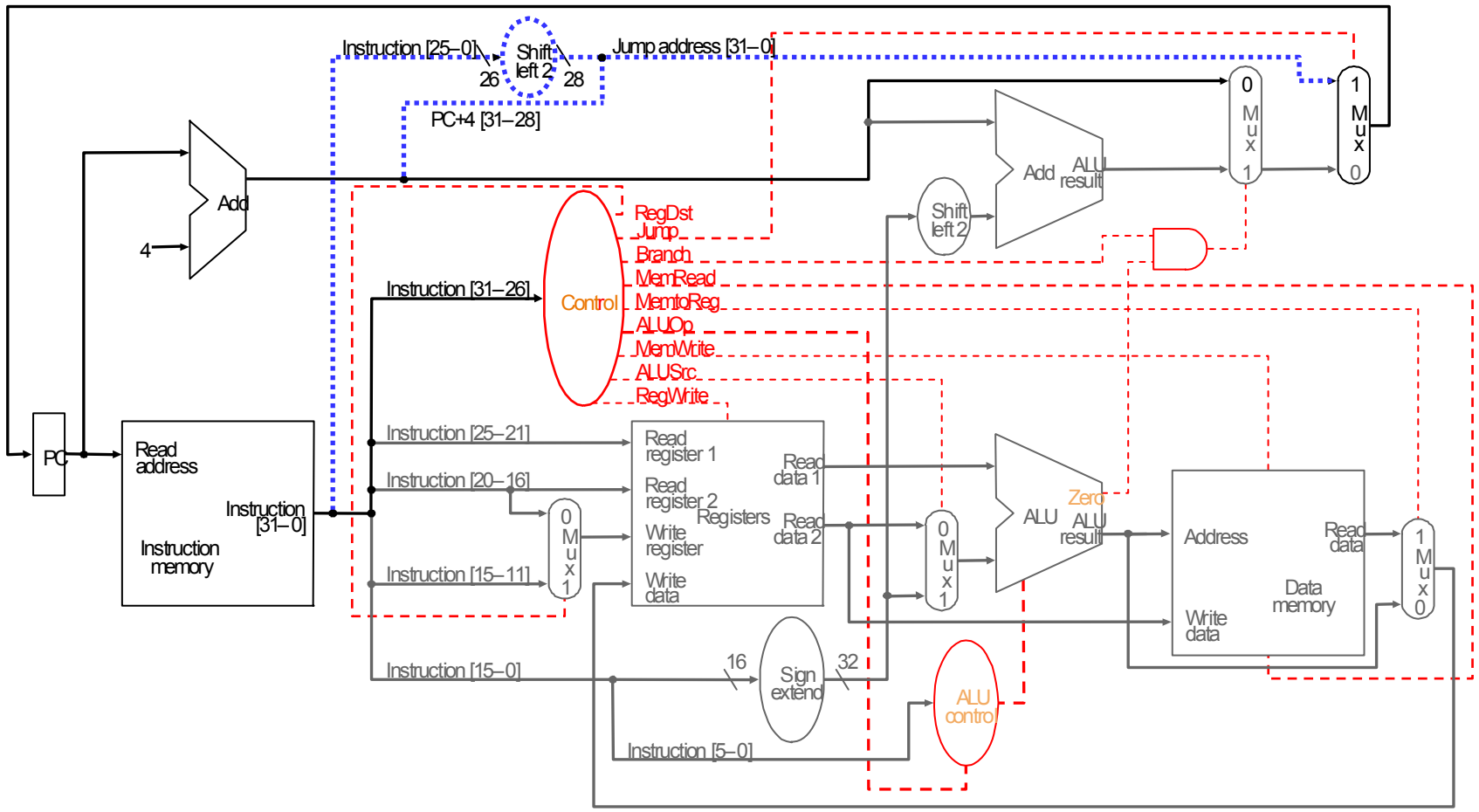
פקודת ה- jump

פירוש הפקודה: J 0101…101111011

הפקודה הבאה נמצאת בכתובת 0101…101111011

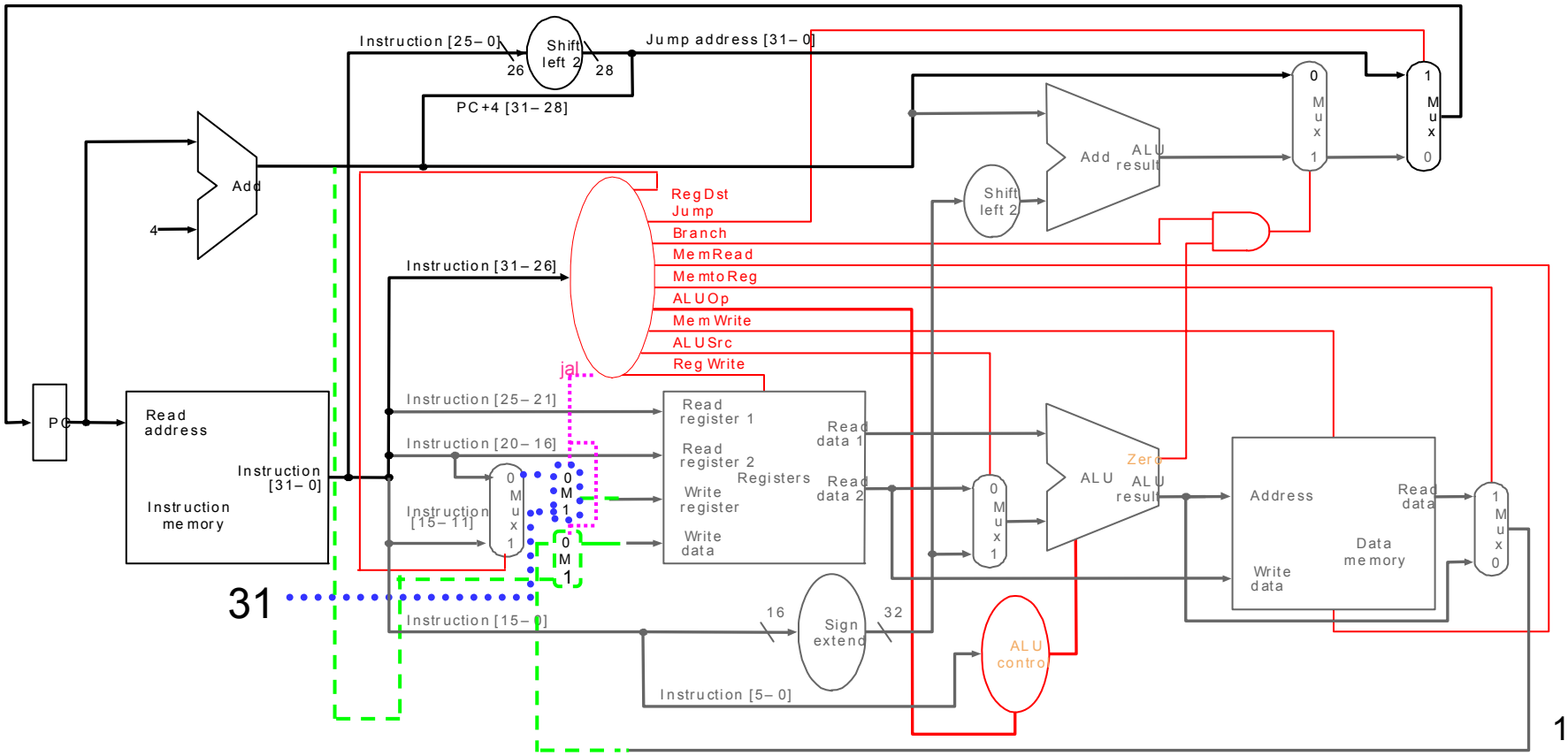


Jump



jal

- הפקודה מורכבת משני חלקים:
  - הכנסת PC+4 לאוגר $ra שמספרו 31
  - פעולת jmp לכתובת שבפקודה



עדכון קווי הבקרה והוספת קו חדש

op	00 0000	00 1000	10 0011	10 1011	00 0100	00 0010	00 0011
	R-type	***addi***	**lw**	**sw**	**beq**	**jump**	**jal**
RegDst	1	0	0	x	x	x	x
ALUSrc	0	1	1	1	0	x	x
MemtoReg	0	0	1	x	x	x	x
RegWrite	1	1	1	0	0	0	1
MemWrite	0	0	0	1	0	0	0
Branch	0	0	0	0	1	x	x
Jump	0	0	0	0	0	1	1
ExtOp	x	0	1	1	x	x	x
ALUop (Symbolic)	"R-type"	Add	Add	Add	Subtract	xxx	xxx
ALUop <1>	1	0	0	0	0	x	x
ALUop <0>	0	0	0	0	1	x	x
jal	0	0	0	0	0	0	1