

תוכן העניינים:

2	מבנה המחשב ותכן לוגי
2	בניית רכיבים לוגים
2	יחידת ה-ALU:
2	סיכום כללי:
7	שאלות:
8	תשובות סופיות:
9	מקבץ האוגרים (Register File):
9	סיכום כללי:

מבנה המחשב ותכן לוגי בניית רכיבים לוגיים

יחידת ה-ALU:

סיכום כללי:

יחידת ה-ALU (Arithmetic Logic Unit):

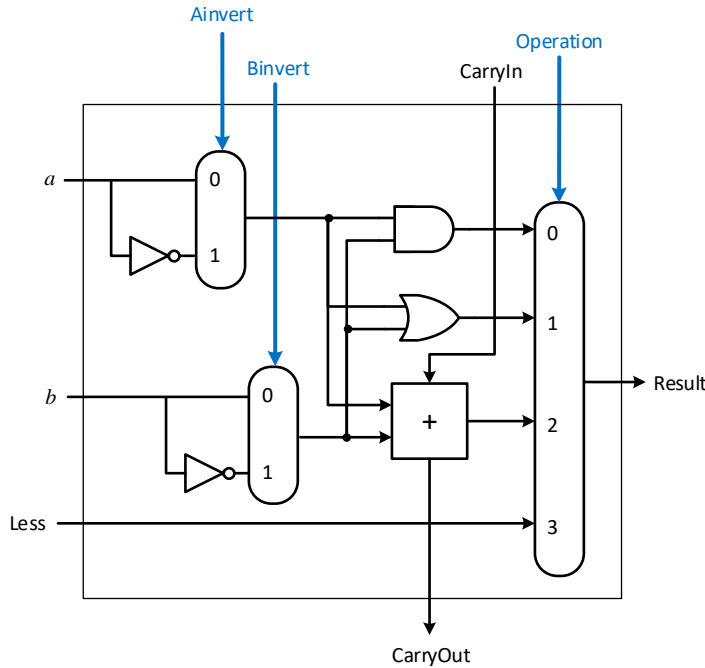
יחידה חישובית המבצעת פעולות אריתמטיות (חיבור וחסור) ולוגיות (פעולות AND, OR וכו').

יחידת ALU של ביט אחד (1-bit ALU):

- פעולות לוגיות AND ו-OR הן מיידידות למימוש:
 - כאשר: $Operation = 0$: $Result = ab$: נקבל
 - כאשר: $Operation = 1$: $Result = a+b$: נקבל
- נממש חיבור באמצעות בלוק F.A.:
 - כאשר: $Operation = 2$: נקבל:
$$CarryOut = a \cdot CarryIn + b \cdot CarryIn + a \cdot b$$

$$Sum = a \cdot \bar{b} \cdot \overline{CarryIn} + \bar{a} \cdot b \cdot \overline{CarryIn} + \bar{a} \cdot \bar{b} \cdot CarryIn + a \cdot b \cdot CarryIn$$
- נממש חיסור ע"י הכנסת $CarryIn = 1$.
 - כאשר: $Operation = 2$ ו- $CarryIn = 1$: נקבל: $Result = a - b$.
- נממש פעולות NOR ו-NAND ע"י שימוש בחוקי דה-מורגן:
 - כאשר: $Operation = 0$, $Ainvert = 1$, $Binvert = 1$: נקבל: $Result = \overline{a+b}$.
 - כאשר: $Operation = 1$, $Ainvert = 1$, $Binvert = 1$: נקבל: $Result = \overline{ab}$.
- נממש פעולת slt כאשר $Operation = 3$.

תיאור סכמתי של יחידת 1-bit ALU:



יחידת ALU של 32 ביטים (32-bit ALU):

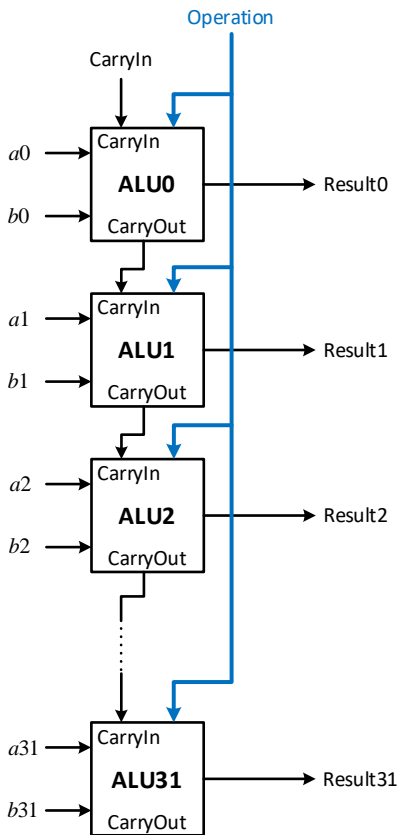
לאחר שהשתמשנו יחידת ALU של ביט אחד, נוכל להתייחס אליו כאל קופסה שחורה ולשרשר מספר יחידות על מנת לממש פעולות על מילים שלמות (32 ביטים).

אופן הסימון:

בהתאם לשירשור, יחידת ה-ALU הנמוכה ביותר בסדר החשיבה תסומן ALU0 ובהתאם כניסותיה ויציאותיה יסומנו כך: A0, B0, CarryIn0, Result0. כשמתקדמים עם סדר החשיבות נקבל סימונים: ALU1 וכו' עד ל-ALU31 עם כל כניסותיו ויציאותיו.

קווי הבקרה:

חשוב לציין כי קווי הבקרה תמיד יהיו זהים לכל יחידות ה-ALU.

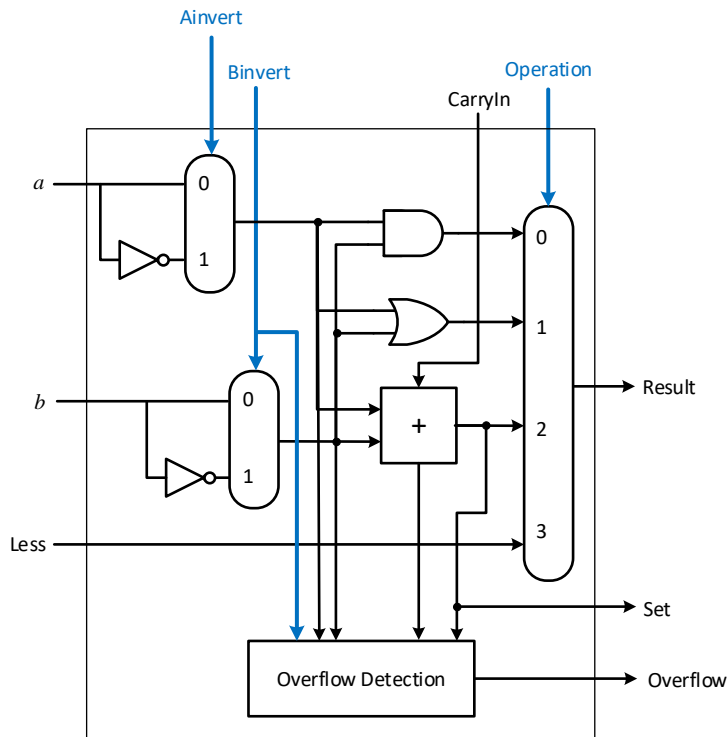


פעולת slt (Let in Less Than) עבור ALU 1-bit:

ביט ה-slt יחזיר ערך של 1 לוגי אם $rs < rt$ וערך של 0 לוגי אחרת (כלומר: $rs \geq rt$).
 בשפת Assembly יש להגדיר אוגר יעד ($rd = \text{Register destination}$) אליו תוצאת ההוראה slt תיכתב. היות ו-slt מחזיר תוצאה של ביט אחד, והוא ה-LSB, הרי שערכו של האוגר הנבחר יהיה 0 או 1.
 במקרה שלא מתקיים $rs < rt$ אז $slt = 0$ ואז התוכן של rd יהיה רצף של 32 אפסים.
 במקרה שמתקיים $rs < rt$ אז $slt = 1$ ואז התוכן של rd יהיה רצף של 31 אפסים וביט ה-LSB שלו יהיה 1.

יחידת ה-ALU האחרונה (ה-MSB):

כאשר $\text{Operation} = 3$ נכנס הערך less בכל אחד מיחידות ה-ALU. בפרט, ב-ALU האחרון נתעניין בתוצאת החיסור $A - B$ ולכן סיבית ה- $\text{sum}[31]$ תשורשר ל-ALU0.

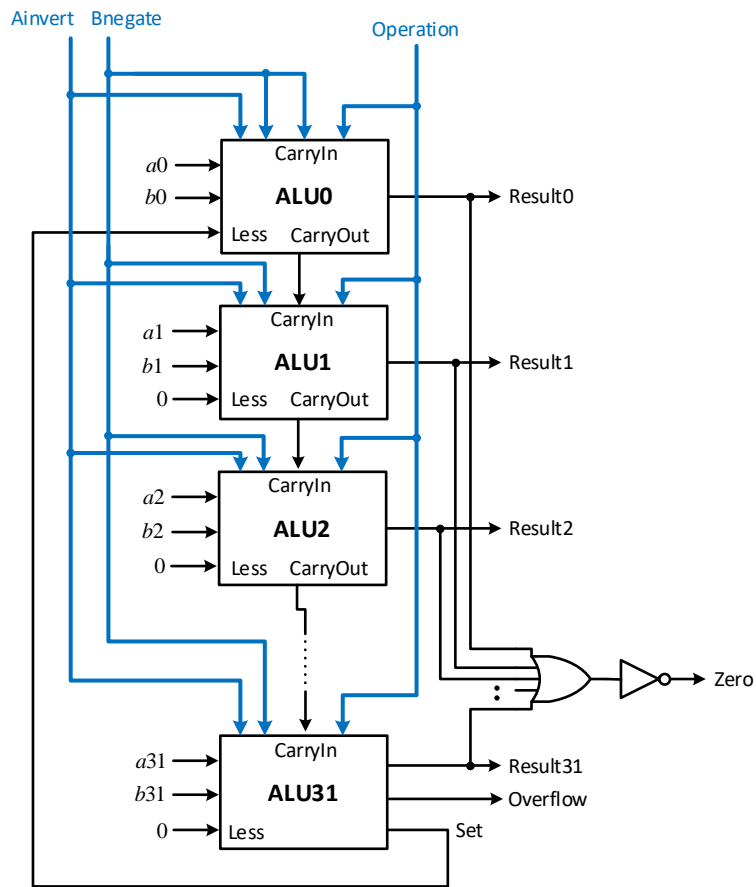


מבנה כללי של יחידת ALU של 32 ביטים:

- מחברים את Binvert ואת CarryIn0 לסיבית בקרה Bnegate.
- מגדירים סיבית בקרה zero לפי:

$$\text{zero} = \overline{(\text{Result}_{31} + \text{Result}_{30} + \text{Result}_{29} + \dots + \text{Result}_1 + \text{Result}_0)}$$

מטרת סיבית זו היא להתריע אם שני המספרים A ו-B זהים או שונים.



יחידת ה-ALU - סימון וטבלאות:

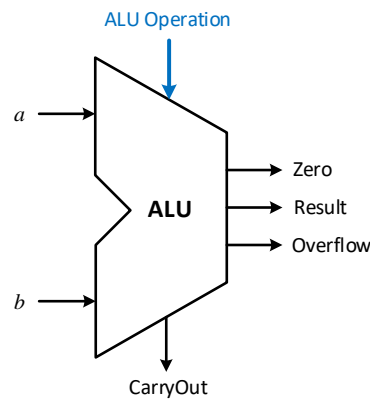
יחידת ALU הינה יחידה המסוגלת לבצע פעולות אריתמטיות (ADD/SUB) ופעולות לוגיות (AND, OR, NOR). ה-ALU מקבל בכניסתו מידע משני אוגרים בני 32 ביטים, A ו-B

$$A = (a_{31} a_{30} \dots a_1 a_0) \text{ ו- } B = (b_{31} b_{30} \dots b_1 b_0)$$

ו-4 קווי בקרה:

MSB		LSB	
ALUOp3	ALUOp2	ALUOp1	ALUOp0
Ainvert	Bnegate	Operation1	Operation0

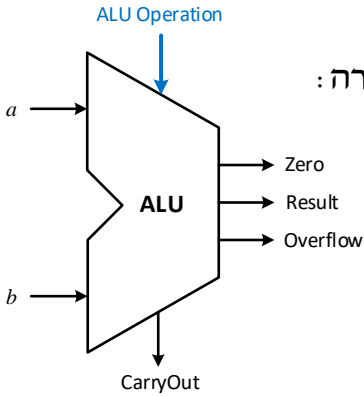
ALU control lines	Function
0000	AND
0001	OR
0010	Add
0110	Subtract
0111	Set on less than
1100	NOR



יחידת ה-ALU מוציאה:

- תוצאה אריתמטית/לוגית: $Result = (Result_{31}, \dots, Result_0)$. כאשר $Result = Operation(A,B)$ והפעולה נקבעת ע"י קווי הבקרה.
- 4 דגלים (ביט בודד) של חיווי:
 - Zero – ביט שנדלק (עולה ל-1) אם המידע בשני האוגרים A ו-B זהה.
 - Carry – הביט $CarryOut_{31}$ שמעיד על גלישה לפי ייצוג ללא סימן.
 - Overflow – ביט שנדלק במידה והתרחשה גלישה אריתמטית.
 - Sign – הביט $Result_{31}$ המעיד כי המספר המאוחסן ב-Result הוא שלילי (1) או חיובי (0) בשיטת המשלים ל-2.

שאלות:



1) בשאלה זו נתייחס למצבי הבקרה של יחידת ALU בסיסית.
א. תארו אלו פעולות תתבצעה ע"י יחידת ALU בכל מקרה:

- 1110 (1)
- 1001 (2)
- 1101 (3)
- 1011 (4)

ב. נניח יחידת ALU של 8 ביטים.

מכניסים את המידע הבא:

$a = 0000\ 1111$ ו- $b = 1111\ 0000$.

קבעו מה צריכים להיות ה-ALU Operation Control Lines עבור:

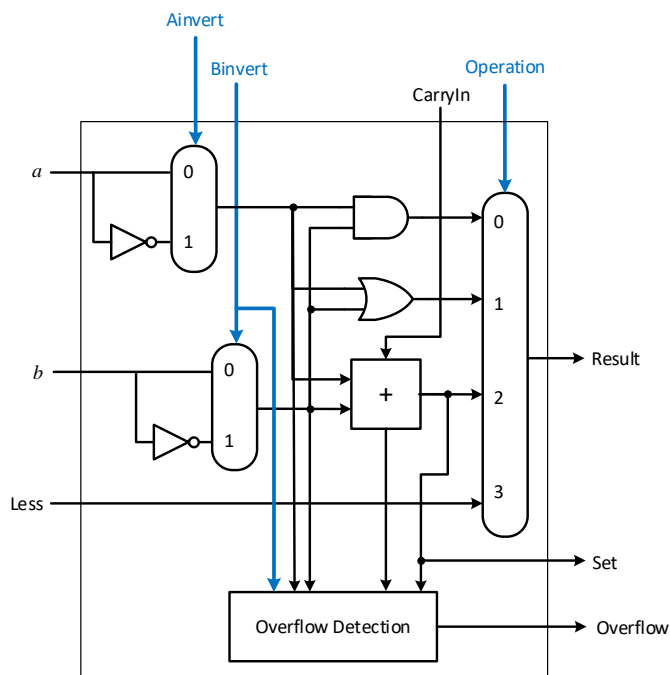
- (1) Result = 0000 0000
- (2) Result = 1111 1111
- (3) Result = 0001 1111

2) בשאלה זו נממש את יחידת ה-Overflow Detection הממוקמת בתוך בלוק ה-ALU האחרון. סיבית ה-Overflow המופיעה בסכמה של יחידת ה-1-bit ALU האחרון, מטרתה להתריע על זליגה אריתמטית. לצורך המימוש, נניח ALU בגודל 32 ביטים.

א. ממשו את יחידת ה-Overflow Detection באמצעות הסיביות sum31 ו-CarryOut31 בלבד.

ב. כעת הניחו כי סיבית ה-CarryOut31 אינה זמינה.

הציעו מימוש נוסף ליחידת ה-Overflow Detection באמצעות הסיביות: sum31, B31, A31.



- 3) לפניכם מספר משפטים.
- קבעו מי מהמשפטים נכון ומי לא נכון. נמקו את טענתכם.
- א. בחיבור של שני מספרים בני N סיביות בייצוג ללא סימן מהצורה:
 $A = [a_{N-1} \dots a_0]$, אם $c_N = 1$ אז הערך העשרוני של סיביות הסכום
 (בלבד) יהיה קטן מערכי המספרים המחוברים.
- ב. סיבית ה-Overflow מתייחסת רק לחיבור של שני מספרים בשיטת
 משלים ל-2 ולא אף הצגה אחרת.
- ג. בייצוג בינארי רגיל של שני מספרים בני N ביטים, סיבית ה-Overflow
 וסיבית ה-CarryOut[N-1] תמיד תהיינה זהות בערכן ובמשמעותן.
- ד. בהינתן יחידת ALU באורך N , זליגה אריתמטית תתרחש רק כאשר
 מחברים שני מספרים שתוצאת ערכם גדולה מדי לייצוג ב- N ביטים.

תשובות סופיות:

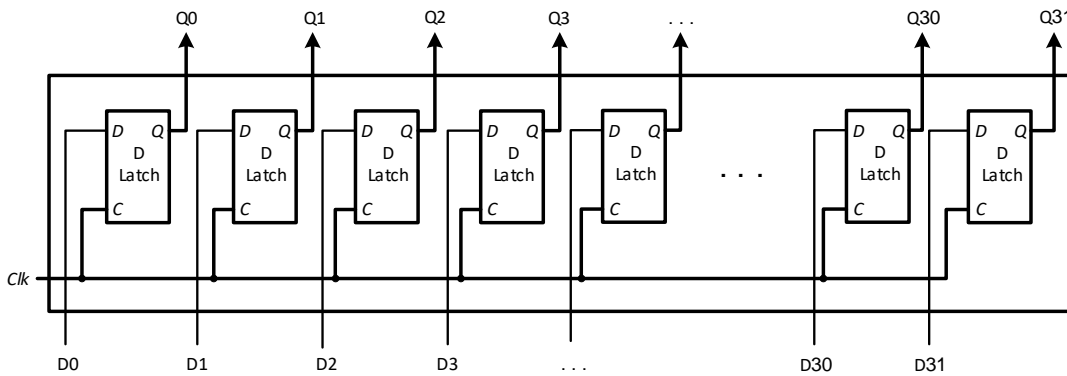
- 1) א. עבור 1110 : $\bar{a} - b = \bar{a} + (\bar{b} + 1)$ עבור 1001 : $\bar{a} + b$
- ב. עבור 1101 : \overline{ab} עבור 1011 : אף פעולה.
- ב. (1) AND(a,b) ב. (2) OR(a,b) ב. (3) Sub(a,b).
- 2) ראו מימושים ושיקולים בסרטון הוידאו.
- 3) א. הטענה נכונה. ב. הטענה נכונה.
- ג. הטענה לא נכונה. ד. הטענה לא נכונה.

מקבץ האוגרים (Register File):

סיכום כללי:

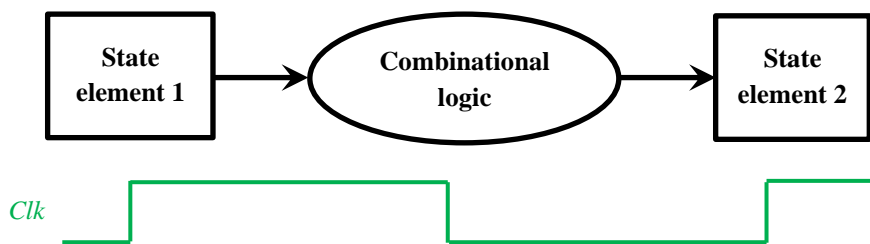
מבנה עקרוני של אוגר/מחסנית (Register):

אוגר (או: מחסנית) בנוי מ-32 יחידות של Edged-Triggered D-latches כמתואר בסכמה הבאה. לכל יחידות הזכרון כניסת שעון משותפת המאפשרת למילה שבכניסת האוגר לחלחל לתוכו. בעליית השעון ערך המילה תישמר במוצאים $Q_{31}Q_{30}\dots Q_1Q_0$.



מתודולוגית תזמון (Clocking Methodology):

כשעוסקים במעגלים הכוללים יחידות צירופיות ורכיבי זיכרון, יש חשיבות לבחירת אופן הפעולה שלהם ביחס למחזור השעון. נתייחס ל-edge-triggered clocking ונתבונן בסכמה הבאה:

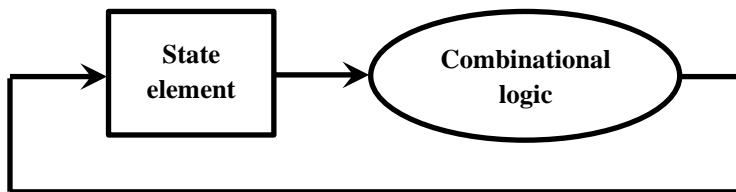


- ערכי הכניסות למעגל הצירופי הן אלו השמורות ב-state element 1 לפני עליית השעון.
- המעגל הצירופי חייב לסיים לבצע את פעילותו כך שיציאותיו יחלחלו ל-state element 2 בטרם עליית השעון ההבאה.
- בעליית השעון הנוספת, מוצאי המעגל הצירופי ייכנס ל-state element 2 ויישמרו שם.

הערה:

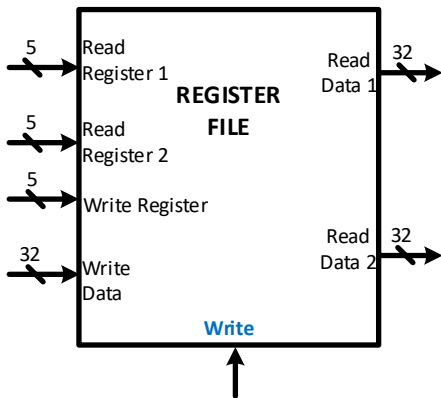
במידה ובכל עליית שעון מתרחשת כתיבה ל-state element, מקובל שלא לכלול את קו הבקרה write. מאידך, אם מתרחשת כתיבה בצורה אסינכרונית, נכלול את קו ה-write לרכיבי הזיכרון וכאשר הוא יהיה על 'גבוה' תתרחש כתיבה (כלומר: עדכון) של מידע בעליית השעון הסמוכה.

מתודולוגית ה-edge-triggered מאפשרת לנו לקרוא את התוכן של אוגר, לשלוח אותו ללוגיקה צירופית כלשהי, ולכתוב אותו חזרה לאותו האוגר:



סכמה של רכיב זיכרון Register Files:

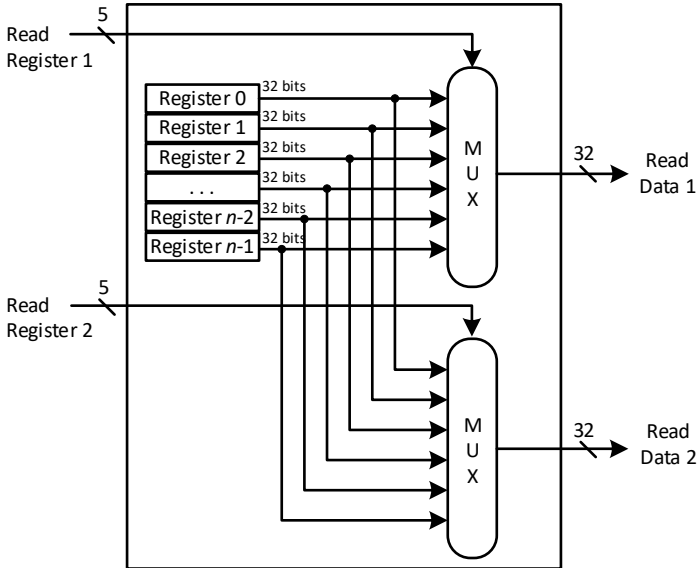
הרכיב ה-Register File של משפחת ה-MIPS32 הוא אחד מאבני הבניין המהותיים ביותר של המעבד. המבנה הבסיסי ביותר כולל:



- שתי כניסות (פורטים) קריאה של מידע מאוגרים
- כניסת כתיבה של מידע
- קו המידע (Write Data BUS line)
- ביט איפשור לכתיבה (Write)
- כניסת שעון לצורך פעולת כתיבה (Clk)
- שתי יציאות מידע מהאוגרים (Read DataX)

מבנה פנימי עקרוני - קריאה מהזיכרון:

קריאה מהזיכרון לא דורשת כניסת שעון מכיוון שהמידע שמור באוגרים בכל זמן. יש לספק רק את מספר האוגר שממנו נרצה לקרוא את המידע.



רכיב הזיכרון של ה-MIPS32 מאפשר קריאה כפולה של מידע מאוגרים:

קריאה לא דורשת כניסת שעון (המידע שמור באוגרים).

בשביל קריאה נצטרך את הכניסות:

- מספר האוגר הראשון.
- מספר האוגר השני.

יצאות הרכיב:

- הערך שבאוגר הראשון.
- הערך שבאוגר השני.

מבנה פנימי עקרוני - כתיבה לזיכרון:

כתיבה לזיכרון מתבצעת על אוגר אחד, אליו יש לכתוב את המידע המבוקש.

חלק זה מורכב מ-3 כניסות:

- מספר האוגר אליו יש לכתוב את המידע (Write Register).
- המידע אותו יש לכתוב (Write Data).
- כניסת שעון (Clk).
- עם ביט אי־פשוט (Write).

