7.11.05

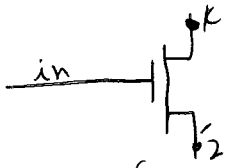
מבוא מתקנים - ארזון

אורי שיש - st.tau.ac.post@orish

שעת קבלה ה' 14-15 מ' 21.

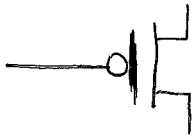
טרנזיסטורים: Transistor Level Logic

(*) טרנזיסטור nFET



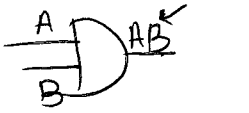
בטרנזיסטור nFET כאשר יש כניסה 1 המעגל סגור (יש הולכה) וכאשר יש כניסה 0 המעגל פתוח (אין כניסה)

(*) טרנזיסטור pFET



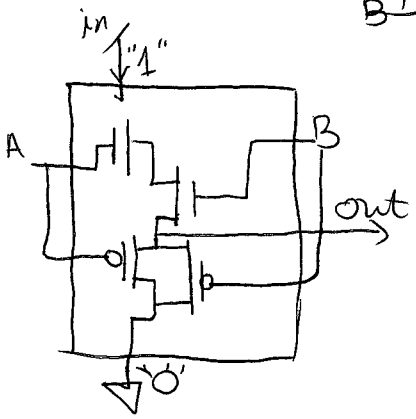
מעגל הפוך מ- nFET

שער AND



A	B	AB
0	0	0
0	1	0
1	0	0
1	1	1

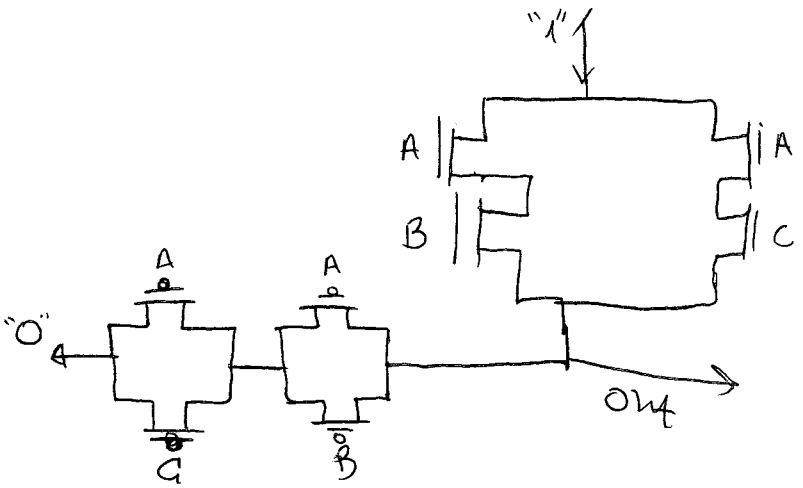
רק כששניהם נכנס כניסה אז יוצא כניסה.



נקבה מעגל שיכול 3 טרנזיסטורים ויתן 1 אמ"מ:

$AB \oplus AC$ + מסמן XOR

XOR ⇐ באמצעות צלמאנים $\begin{cases} A \text{ ו- } B \\ A \text{ ו- } C \end{cases}$



נוכיח כי $AB+AC=A(B+C)$ לפי טבלת אמת:

A	B	C	AB	AC	B+C	A(B+C)	AB+AC
0	0	0	0	0	0	0	0
0	0	1	0	0	1	0	0
0	1	0	0	0	1	0	0
0	1	1	0	0	1	0	0
1	0	0	0	0	0	0	0
1	0	1	0	1	1	1	1
1	1	0	1	0	1	1	1
1	1	1	1	1	1	1	1

הטור הראשון זהה

חישוב בסיסים:

$(123)_{10}=(\ ?\ )_2$

	/2
123	1
61	1
30	0
15	1
7	1
3	1
1	1
0	

התשובה אמורה להיות כך:

1111011

$2^6\ 2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0$

כפי שאבר הפוך:

$(1111011)_2=$

$2^0+2^1+2^3+2^4+2^5+2^6=(123)_{10}$

כאשר יש שבר אשרונים:

$2^1\ 2^0\ 2^{-1}\ 2^{-2}\ 2^{-3}$

$(11.011)_2$

$=2^1+2^0+2^{-2}+2^{-3}=3\frac{3}{8}$

(*) $(11.1111\ldots)_2=4$

	x2	
0.375	0.75	$(0.011)_2$
0.75	1.5	
0.5	1.0	
0		

בסיס 8 (אוקטלי)

$8^2 \quad 8^1 \quad 8^0$

$(345)_8 = 3\cdot 8^2 + 4\cdot 8^1 + 5\cdot 8^0 = (229)_{10}$

$(345)_8 = (\underbrace{011}_{3}\,\underbrace{100}_{4}\,\underbrace{101}_{5})_2$

(*) במעבר מבסיס 2 לבסיס "פלאים" כל שלשה מספרים בין אחרים. במעבר הפוך עושים כך גם כן, ואם צריך "מרפדים" באפסים.

בסיס 16 (הקסדצימלי)

16 מספרים: $0,1,\dots,9,A,B,C,D,E,F$

במעבר בין מערכת מבסיס 2 ל"ז קיבוץ מרבעיות.

$(\underbrace{1110}_{E=15}\,\underbrace{0101}_{5})_2 = (E5)_{16}$

מעברים שברים

הזרים מהצד כפי חולו!

$(1101.1011101)_2$

$(1\ 6\ D\ .\ A\ D)_{16}$

$(\underbrace{101}_{5}\,\underbrace{101}_{5}\,\underbrace{101}_{5}.\underbrace{101}_{5}\,\underbrace{011}_{3}\,\underbrace{010}_{2})_8$

14.11.2005

יצוג מספרים

1. הופכים כל 0 ל-1 וההפך.
2. מחברים "1" למספר.

דוגמא: $-103+(-41)=$

$|103| = 01100111 \Rightarrow 10011000 \Rightarrow 10011001 = -103$

$|41| = 00101001 \Rightarrow 11010110 \Rightarrow 11010111 = -41$

Carry:

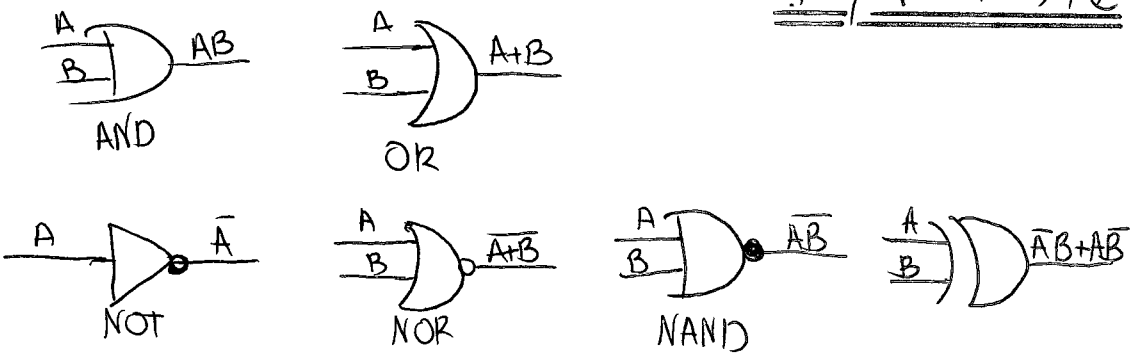
$$\begin{array}{r} 10011001 \\ +\ 11010111 \\ \hline 01100000 \end{array} = 96$$

(*) הבעיה יכולה שלא מכוון שהבעיה

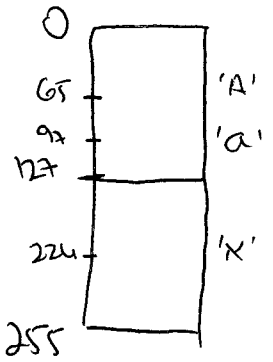
בגלל 8 ביטים אפשר בין $-128,\dots,127$

לכן מתקבל רק כאשר $C_7 \neq C_6$

שערים לוגיים



קוד אסקי



מקודד את הטקסט

המחרוזת $0,\dots,255$

$'A'=65,\ 'B'=66,\dots$

$'a'=97,\dots$

תקן ISO8859

משתמש בהוספה שפה נוספת ותקן אסקי כך שהתווי

אסקי בחצי $0,\dots,127$ השאר כדים ו-128 ומעלה

הסוף של אווים של השפה

תקן ISO 10646

מכיל 0x10FFF ....,0

שמכיל 2^{20} תווים (מכל השפות, כל הסוגים)

תקן Unicode - פומה וקפה, קבע יותר מתקנים וכוללני

תקן UTF - 8 ברור מה זה.

ISO-VISUAL

כתב שפות שמימין לשמאל הפיר, יעני שום

יכתב פ-ו-ר-ש.

21.11.2005

מבוא למחשבים - תרגול

חוקי דה-מורגן:

(*) $\overline{AB} = \bar{A} + \bar{B}$

(*) $\overline{A+B} = \bar{A} \cdot \bar{B}$

פונקציה:

$$\overline{(x\bar{y}+w\bar{z})(w\bar{x}+y\bar{z})} =$$

$$\overline{(x\bar{y}+w\bar{z})} + \overline{(w\bar{x}+y\bar{z})} =$$

$$\overline{x\bar{y}} \cdot \overline{w\bar{z}} + \overline{w\bar{x}} \cdot \overline{y\bar{z}} =$$

$$(\bar{x}+y)(\bar{w}+z)+(\bar{w}+x)(\bar{y}+z) =$$

$$\bar{x}\bar{w}+\bar{x}z+y\bar{w}+zy+\bar{w}\bar{y}+\bar{w}z+x\bar{y}+xz =$$

$$\bar{x}z+xz+\bar{w}y+\bar{w}\bar{y}+\bar{w}z+\bar{x}\bar{w}+yz+x\bar{y} =$$

$$z(x+\bar{x})+\bar{w}(y+\bar{y})+\bar{w}z+\bar{x}\bar{w}+yz+x\bar{y} =$$

$$z+\bar{w}+\bar{w}z+\bar{x}\bar{w}+yz+\bar{y}x =$$

$$z+\bar{w}(1+z)+\bar{x}\bar{w}+yz+\bar{y}x =$$

$$z+\bar{w}+x\bar{y}$$

חוקים שהשתמשנו בהם: $A+AB = A(1+B)=A$

$A\bar{B}+AB = A(\bar{B}+B)=A$

מערכת פונקציונלית:

מערכת פונקציונלית $\Leftrightarrow$ ניתן ובנוי בעזרתה כל פונק' אחרת. מערכות פונקציונליות נפוצות:

(*) $\{AB, \bar{A}\}$

(*) $\{A+B, \bar{A}\}$

(*) $\{\overline{AB}\}$

(*) $\{\overline{A+B}\}$

(1) נתון פונ' $f(x,y,z)=\bar{x}\bar{y}+\bar{x}\bar{z}+\bar{y}\bar{z}$

הראה ש- $\{f,1\}$ אוניברסלי:

$f(x,y,1)=\bar{x}\bar{y}+0+0=\overline{x+y}$ אוניברסלי! $\Leftarrow$ NOR

(2) נתון פונ' $f(x,y,z)=\bar{x}y\bar{z}+\bar{y}\bar{z}+x\bar{y}$

האם $\{f\}$ אוניברסלי?

נחלב: $f(x,x,z)=\bar{x}x\bar{z}+\bar{x}\bar{z}+\bar{z}x\bar{x}=$

$0+\bar{x}\bar{z}+0=\overline{x+z}=NOR$ אוניברסלי! $\Leftarrow$

(3) נתונים הפונ' $f_2(x,y)=x\bar{y}$; $f_3(x,y)=x+\bar{y}$

הראה ש- $\{f_2,f_3\}$ אוניב'.

$f_2(x,x)=x\cdot\bar{x}=0$

$f_3(0,y)=0+\bar{y}=\bar{y} \Rightarrow f_3(f_2(y,y),y)=\bar{y}$

$f_2(x,\bar{y})=xy \Rightarrow f_2(x,f_3(f_2(y,y),y))=xy$

קיבלנו מערכת $\{AND,NOT\}$ $\Leftarrow$ שהיא מע' אוניברסלי!

נתון: $f_1(x,y,z)=\sum(0,2,4,6,7)$

המשמעות:

$f_1=1 \Leftrightarrow$ בערכים

בשורה i ...

(*) שים ♥ בערך הזה

$f(y,x,z)\neq f(x,y,z)$!

	x	y	z	f_1
0	0	0	0	1
1	0	0	1	0
2	0	1	0	1
3	0	1	1	0
4	1	0	0	1
5	1	0	1	0
6	1	1	0	1
7	1	1	1	1

הפונקציה שמתאימה לכן תהיה: (SOD)

$$\bar{x}\bar{y}\bar{z}+\bar{x}y\bar{z}+x\bar{y}\bar{z}+xy\bar{z}+xyz$$

נקבל את f_1 באמצעות $\{f_1,f_2\}$ (כי זו מערכת אוניברסלית!) ולכן עלינו לעשות כמה שיותר פישוט:

$$=\bar{z}(\bar{x}\bar{y}+\bar{x}y+x\bar{y}+xy)+xyz$$

$$=\bar{z}+xyz$$

(סעיף ד) הנחה $A+\bar{A}B=A+B$ ולכן:

$$=\bar{z}+xy$$

$$=f_3(xy,z)$$

$$=f_3(\underbrace{f_2(x,f_3(f_2(y,y),y))}_{xy},z)$$

דרך נוספת להציג f_1: במכפלה של סכומים (POS)

(לא SOD)

$$f_1=\overset{1}{(x+y+\bar{z})}\overset{3}{(x+\bar{y}+\bar{z})}\overset{5}{(\bar{x}+\bar{y}+\bar{z})}$$

הסימון יהיה: $\Pi(1,3,5)$

ולכן כנ"ל: $\Sigma(0,2,4,6,7)=\Pi(1,3,5)$

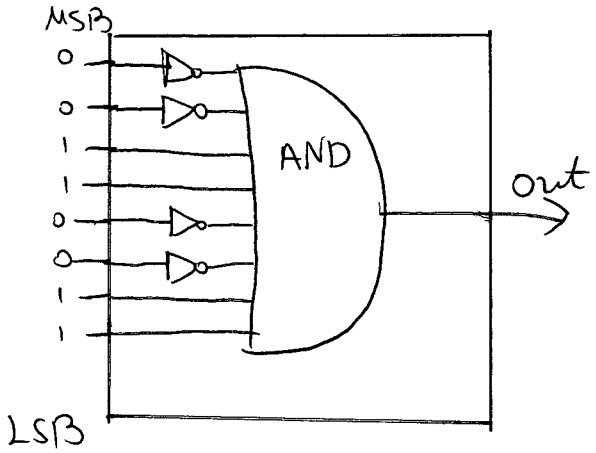
בנה מעגל שעבור רצף ASCII של אות גדולה L

אם האות היא C

ערך אסקי של 'C' = 48

ערך אסקי של 'C' = 51

בבינארי: 00110011



28.11.2005

מבוא מחשבים

מפות קרנו.

$f(x,y,z) = \sum(0,2,4,6,7)$

z \ xy	00	01	11	10
0	1 (0)	1 (2)	1 (6)	1 (4)
1	(1)	(3)	1 (7)	(5)

המטרה: כיסוי בבלוקים את ה-1-ים.

1. הבלוקים יכסו את כל ה-1-ים.
2. הבלוקים בגודל חזקת 2.
3. הבלוקים יהיו גדולים ככל הניתן.
4. מותר חפיפה ואפילו רצוי.
5. הכלה גרועה.

הבלוק הראשון - נקבוע מה משותף בכולם: $z=0$ כלומר, הביט הכי פחות $\bar{z}$. הבלוק השני הוא xy. סה"כ יש לנו:

$\bar{z}+xy$.

$f(w,x,y,z) = \sum(4,5,8,12,13,14,15)$

בוקמאפ

yz \ wx	00	01	11	10
00	(0)	1 (4)	1 (12)	1 (8)
01	(1)	1 (5)	1 (13)	(9)
11	(3)	(7)	1 (15)	(11)
10	(2)	(6)	1 (14)	(10)

$\Rightarrow wx + x\bar{y} + w\bar{y}\bar{z}$

$wx \Downarrow$ הכור, $x\bar{y} \Downarrow$ הרבע, $w\bar{y}\bar{z} \Downarrow$ הכלא

$f(A,B,C,D) = \pi(0,5,7,13,14,15)$

CD \ AB	00	01	11	10
00	0 (0)	(4)	(12)	(8)
01	(1)	0 (5)	0 (13)	(9)
11	(3)	0 (7)	0 (15)	(11)
10	(2)	(6)	0 (14)	(10)

$f = (A+B+C+D)(\bar{B}+\bar{D})(\bar{A}+\bar{B}+\bar{C})$

$f(A,B,C,D,E)=\sum(0,2,5,8,13,15,16,18,21,24,29,31)$

DE \ ABC	000	001	011	010	110	111	101	100
00	1 (0)	(4)	(12)	1 (8)	1 (24)	(28)	(20)	1 (16)
01	(1)	1 (5)	1 (13)	(9)	(25)	1 (29)	1 (21)	(17)
11	(3)	(7)	1 (15)	(11)	(27)	1 (31)	(23)	(19)
10	1 (2)	(6)	(14)	(10)	(26)	(30)	(22)	1 (18)

שאלה!

$f(a,b,c,d)=\sum(5,6,13)+\sum_{\phi}(0,1)$

$f_1(a,b,c,d)=\sum(0,1,5,6,13,8,9)$

מצא f_2 כך ש: $f=f_1 \cdot f_2$

CD \ AB	00	01	11	10
00	ϕ (0)	(4)	(12)	(8)
01	ϕ (1)	1 (5)	1 (13)	(9)
11	(3)	(7)	(15)	(11)
10	(2)	1 (6)	(14)	(10)

=

f_1

1			1
1	1	1	1
	1		

f_2

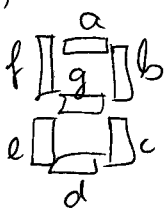
ϕ	ϕ	ϕ	0
ϕ	1	1	0
ϕ	ϕ	ϕ	ϕ
ϕ	1	ϕ	ϕ

אם כך (עכ) כי $f_2=f$ (שמנייה במרכז)

ולכן עכ $f_2=f$.

7 Segment

מייצג את הנורות של המספרים, מקבל רק 4 ביטים (מספרים 0,...,9) ומחזיר מין של a,b,c,d,e,f,g שיבוקש.



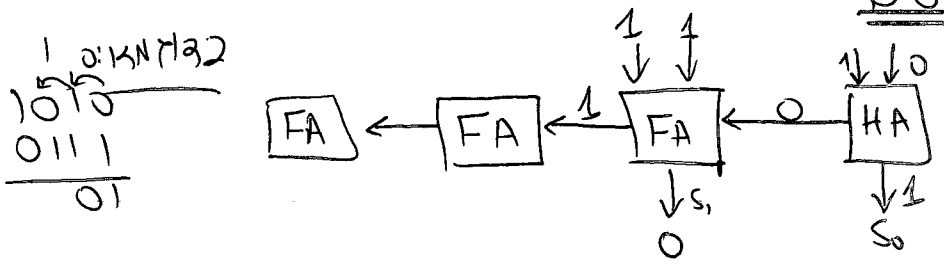
I_3	I_2	I_1	I_0	g
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	ϕ
1	0	1	1	ϕ
1	ϕ	0	0	ϕ
1	1	0	1	ϕ
1	1	1	0	ϕ

מספרים 2,3,4,5,6,8,9 שהם נדלקים.

5.12.2005

מבנה מחשבים - ארז'ן

מחבר 4 ביטים

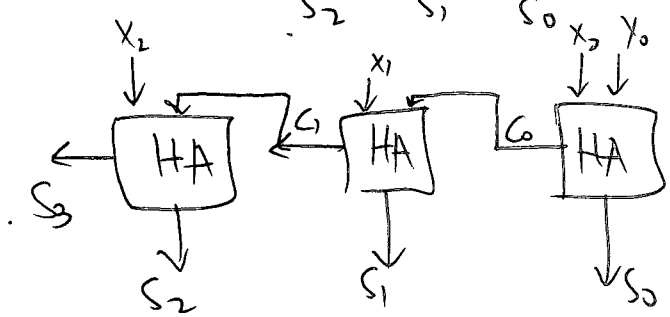


HA - Half Adder מחבר בלי Carry.

FA - Full Adder מחבר עם Carry.

(*) בכל מעגל המחבר מס' בן 3 ספרות עם' בן ספרה אחת, נעזרנו ביחידות HA בלבד

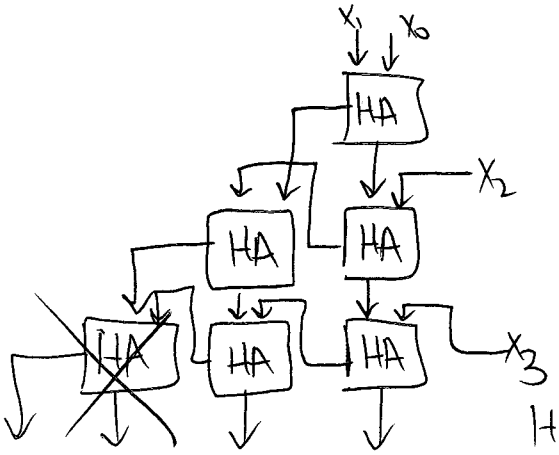
$$\begin{array}{r} X_2 \quad X_1 \quad X_0 \\ + \quad X_0 \\ \hline S_2 \quad S_1 \quad S_0 \end{array}$$



(*) כמה "ו"-ים יש במס' ? נעזרים רק HA.

גודל המס' $X_0 X_1 X_2 \ldots X_{n-2} X_{n-1}$

הפלט: $X_0 + X_1 + \ldots + X_{n-1}$



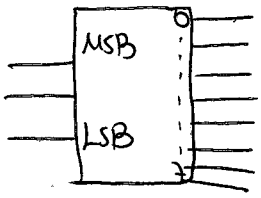
שים ♥!

יש בעיקרה הזרוע

בגודל אין צורך ב- HA

הנותרים רק אפשר לחסוך.

Decoder 3x8

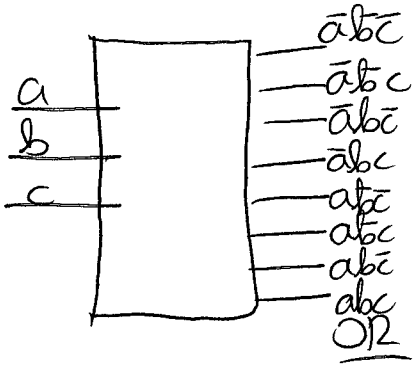


פעולת "הדיקודר" שקול מס' בינארי ומדליק את הקו שמתאים לו בדיוק.

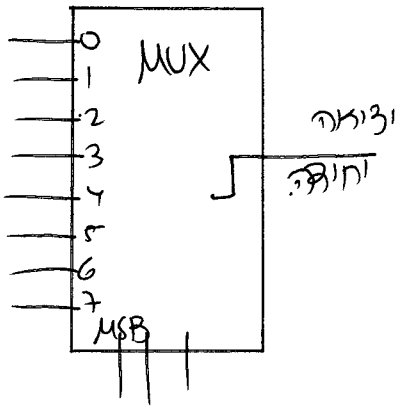
(*) ממש את הפונ' $f(a,b,c)=ab+bc$ באמצעות Dec 3x8 ונייצג את הפונ'

$$ab+bc=abc+ab\bar{c}+bc\cdot\bar{a}+bca$$

כלומר, שם היציאות $abc(7)$, $ab\bar{c}(6)$, $\bar{a}bc(3)$ ונשים בשער OR



מרבב MUX



בדוגמא ל- MUX 8x1 יש 8 נכי כניסה אחד מהם נבחר ליציאה.

(*) ארגון: ממש את הפונ' ע"י MUX ובדוגמה:

$$f(x,y,z,w)=\sum(0,1,5,7,13,14)$$

שממש ב- MUX 16x1 בכניסות 0,1,5,7,13,14 כניסה 1 יוזן ובשאר 0 יוזן. כך כשהבורר שעליו יבין איזה מהם אז הוא יהיה ביציאה ולכן יהיה 1 יוזן.

ניתן לממש פונקציה זו בעזרת MUX 8×1 ושער NOT.

x	y	z	w	F
0	0	0	0	1
0	0	0	1	1
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

לכל שילוב
הכניסות קבוע

$00 \Rightarrow 0$

$01 \Rightarrow w$

$10 \Rightarrow \bar{w}$

$11 \Rightarrow 1$

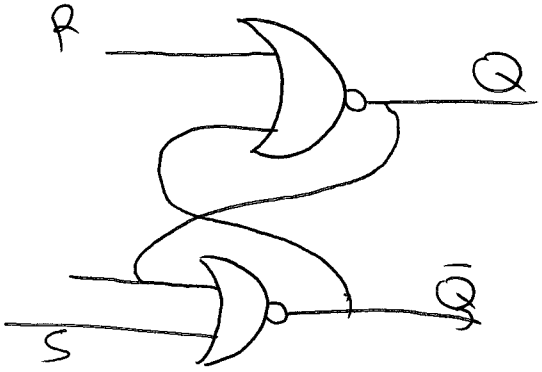
```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY FullAdder IS
  PORT ( Cin, x, y : IN std_logic;
         S, Cout   : OUT std_logic);
END FullAdder;

ARCHITECTURE LogicFunc OF FullAdder IS
BEGIN
  s <= x xor y xor Cin after 5 ns;
  Cout <= (x and y) or (x and Cin) or (y and Cin) after 5 ns;
END LogicFunc;


LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE WORK.FullAdder;

ENTITY  Adder4 IS
     PORT (a, b: IN std_logic_vector(3 downto 0);
        Cin : IN std_logic;
        sum: OUT std_logic_vector (3 downto 0);
        Cout, V: OUT std_logic);
     END Adder4;

architecture fouradder_structure OF Adder4 IS
  SIGNAL c: std_logic_vector (4 downto 0);
  COMPONENT FullAdder
    PORT ( Cin, x, y : IN std_logic;
           S, Cout   : OUT std_logic);
  END COMPONENT;
BEGIN
     FA0: FullAdder
       PORT MAP (Cin, a(0), b(0), sum(0), c(1));
     FA1: FullAdder
       PORT MAP (C(1), a(1), b(1), sum(1), c(2));
     FA2: FullAdder
       PORT MAP (C(2), a(2), b(2), sum(2), c(3));
     FA3: FullAdder
       PORT MAP (C(3), a(3), b(3), sum(3), c(4));
     V <= c(3) xor c(4);
     Cout <= c(4);
END fouradder_structure;
```

```
ENTITY TestAdd IS
END TestAdd;

LIBRARY std;
USE std.textio.all;
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;
USE ieee.std_logic_textio.all;
USE WORK.Adder4;

ARCHITECTURE Test OF TestAdd IS
  COMPONENT Adder4
    PORT (a, b: IN std_logic_vector(3 downto 0);
        Cin : IN std_logic;
        sum: OUT std_logic_vector (3 downto 0);
        Cout, V: OUT std_logic);
  END COMPONENT;

  SIGNAL AAAA, BBBB, SSSS: std_logic_vector (3 downto 0);
  SIGNAL cin_1, cout_1, V_1: std_logic;

BEGIN
  T1: Adder4 PORT MAP(AAAA,BBBB,cin_1,SSSS,cout_1,V_1);
  p: PROCESS IS
    VARIABLE my_line : line;
  BEGIN
    cin_1<='0';
    AAAA<="1110";
    BBBB<="0101";
    WAIT FOR 40 ns;
    write(my_line, string'("sum = "), left);
    write(my_line, cout_1, left, 1);
    write(my_line, SSSS, left, 10);
    writeline(output, my_line);

  END PROCESS;
END Test;
```

12.12.2005

מבוא מחשבים - תרגול

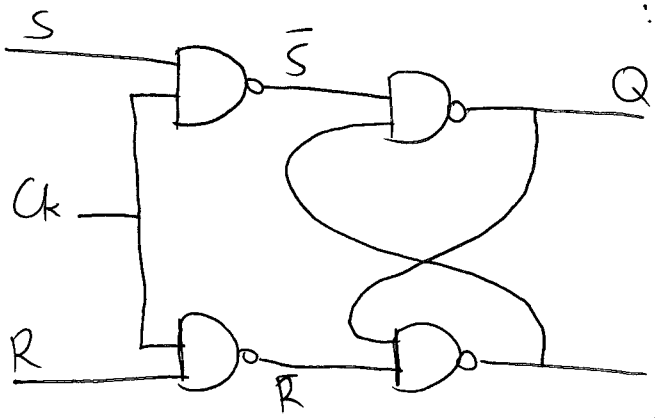
SR-Latch



מה שמיוחד בלבה כזה
ש- Q שומר את הביט.
אם נרצה שישמור אפס
נכניס 1 ל- Reset
1-0 ל- set

אחרת נכניס 0 ! 1 ל- set ואז ישמר אותה.
אחרת אם יכנס 0,0 אז יהיה פלים ל- 1,1 כך ואז
טוב כי כל הזמן יקפוץ.

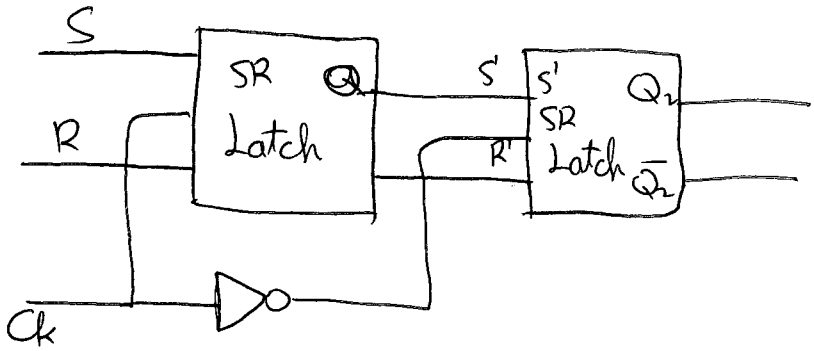
מימוש בשערי NAND:



~~כך יאמר יוג'ב כי~~
~~אם פלים נכניס 1,1~~
~~כך אז יהיה בל"א.~~

Ck - שעון, ~~אם~~
שמאפשר מין disable\Enable

SR-Flip Flop



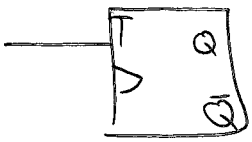
S	R	Q_{t+1}
0	0	Q_t
0	1	0
1	0	1
1	1	אז מוקל!

D-Flip Flop

D	Q_{t+1}
0	0
1	1

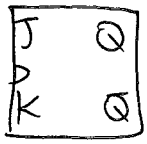
T-Flip Flop

T	Q_{t+1}
0	Q_t
1	$\overline{Q_t}$

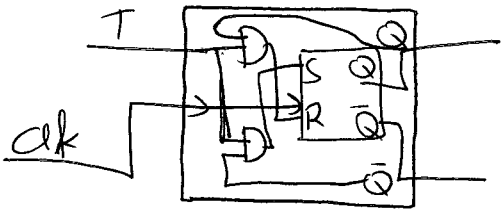


JK- Flip Flop

J	K	Q_{t+1}
0	0	Q_t
0	1	0
1	0	1
1	1	$\overline{Q_t}$



(*) מממש T-FF ע"י SR-FF

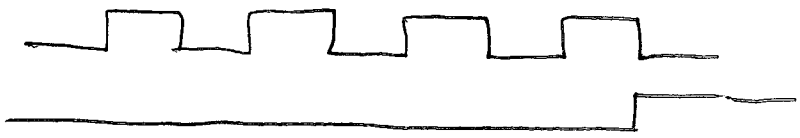


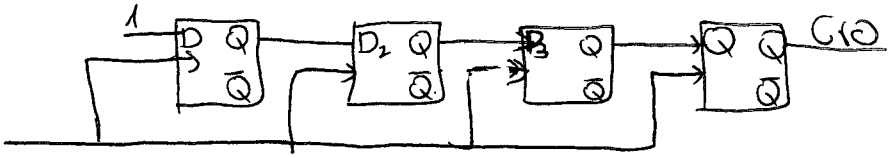
T	Q	S	R
0	0	0	∅
0	1	∅	0
1	0	1	0
1	1	0	1

$S = T\overline{Q}$

$R = TQ$

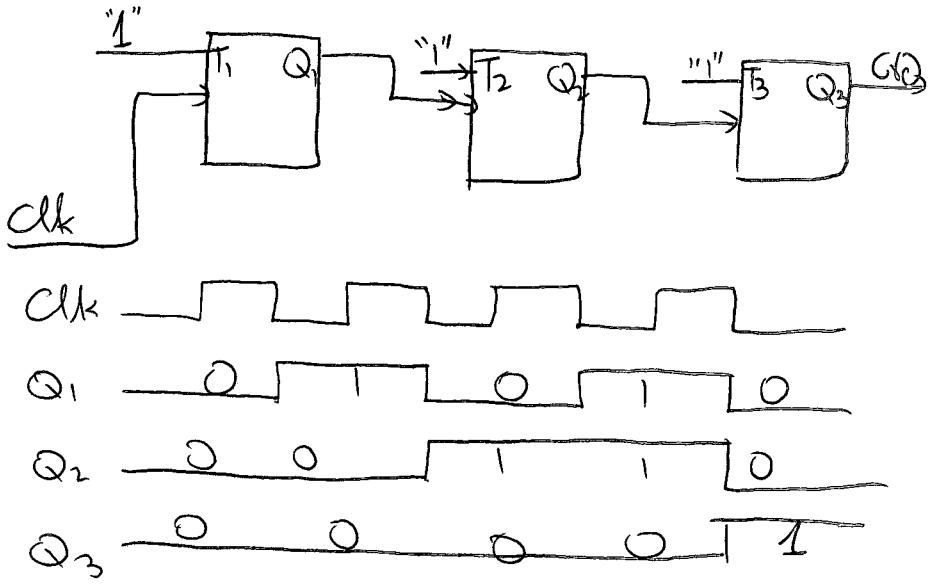
(*) נאון שעון. קח מעגל ספר קל. פלט אחד שהופך מ-0 ל-1 אחרי 4 מחזורי שעון.





שים ♥ ניתן כי כל FF מאותחל ל-0.

מונה עם 3 יחידות FF



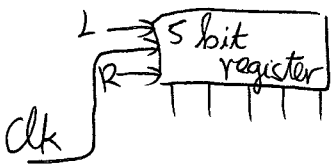
אם נקבל אחרי 3½ מחזורים נפרוץ $Q_1 \cdot Q_2 \cdot Clk$

שים ● בדגש שהשעון כן פועל מעכב אבל בפועל נוצרים "פיקים" - אכן גם יגרום בהרבה יחידות.

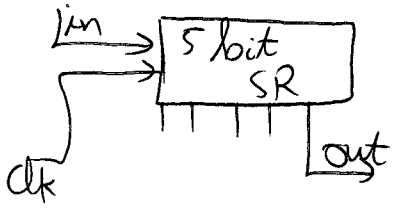
19.12.2005

מבנה מחשבים.

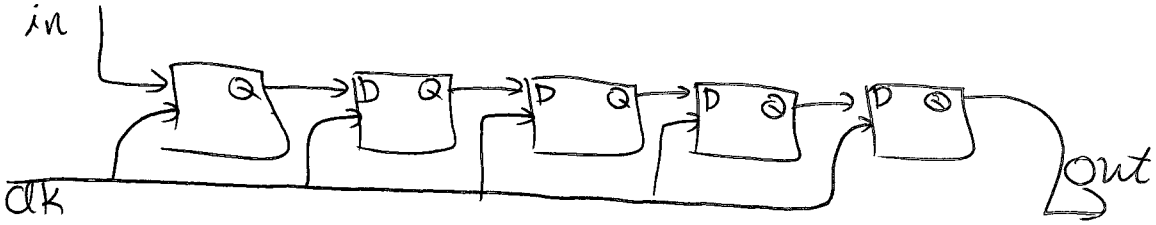
מאגרים - Registers



5 bit Shift Register



במבנה הזה כל מחזור שעון
יוצא ב- out ביט אחד
ונכנס ב- in ביט חדש

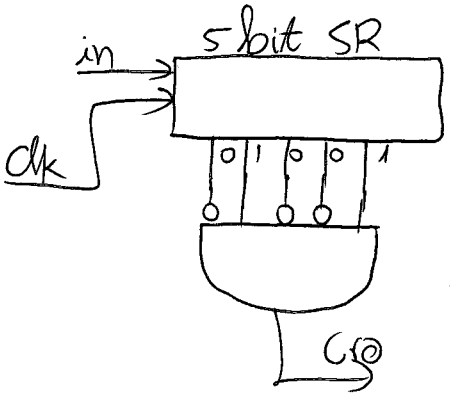


מינימום של Register

DFF 5 יש

דוגמה:

כל זמן מגיע שמגיע מספר מן זוגי פרט
פרט שביניהם לא הרצף 0100
עדיין יש הרבה: 00000100111 התוצאה:
00000010000



שים ♥
הרבה יבבוך
בהיסוך כי
כך מי מספר
הוא יבבה

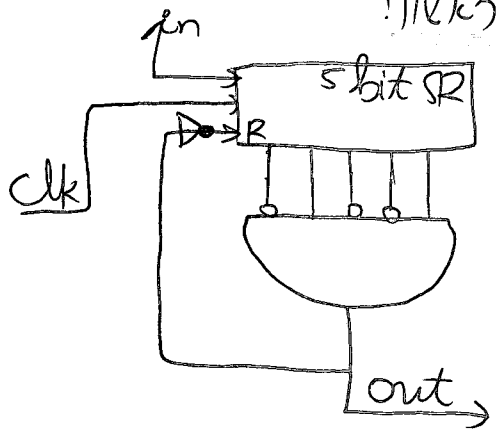
נעבור לבעיה המעניינת: עכשיו אנו מוצאים הבעיה, כך שמסיבות עצ ותקלות.

כאומר על הרגע 1100100100011

הסטט יהיה: 00000100000

כי השני "חלקי" ראשון.

סרטון ראשון:



בסרטון כה ברור שמתקבל הרגל עושים Reset

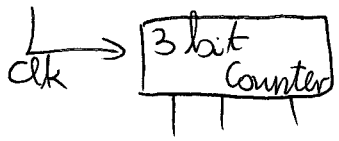
כאומר מוחקים כל הנתונים ומתחילים מהתחלה (Reset מקבל 0 עכשיו, אם נשים מספר)

בעיות בשיטה:

(*) ה- Reset מתחיל רק בסדרת השרון הבאה ובמקום כה המשמש עושי מקום מ "מספס"

(*) אם היינו מתפשרים אומאו סמא היה אנו בעיה כי אם האופס כל הביטים יכבו 0 והיו מספרים 4 מחבר בשבל עקבי.

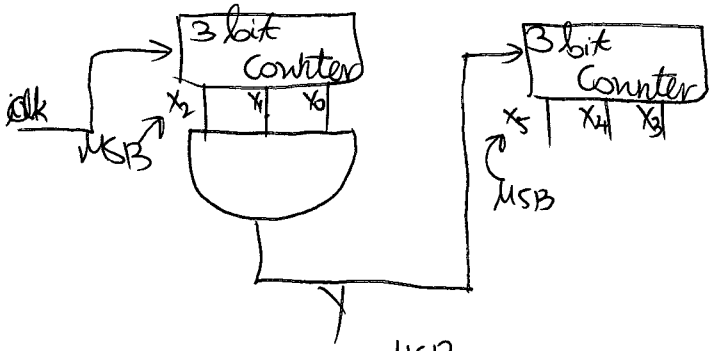
סרטון עבודה: ברור שקיבלנו מחבר מדוע נוסף שמוציא 0 במשך 5 סדימות ומחבר עשר AND



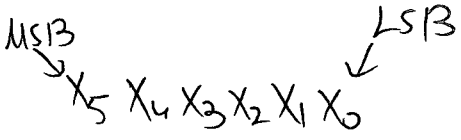
3 bit Counter

הרכיב הנ"ל סופר רק 8.
בכל פעימת שעון המספר המס' שבמונה דולג ב-1
מ-8 הוא יעבור 0-0 ומחדש מחדש.

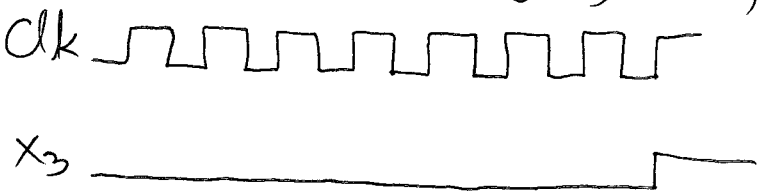
תרגיל: לחבר 2 יח' counter 3-bit.
קבל counter 6 bit



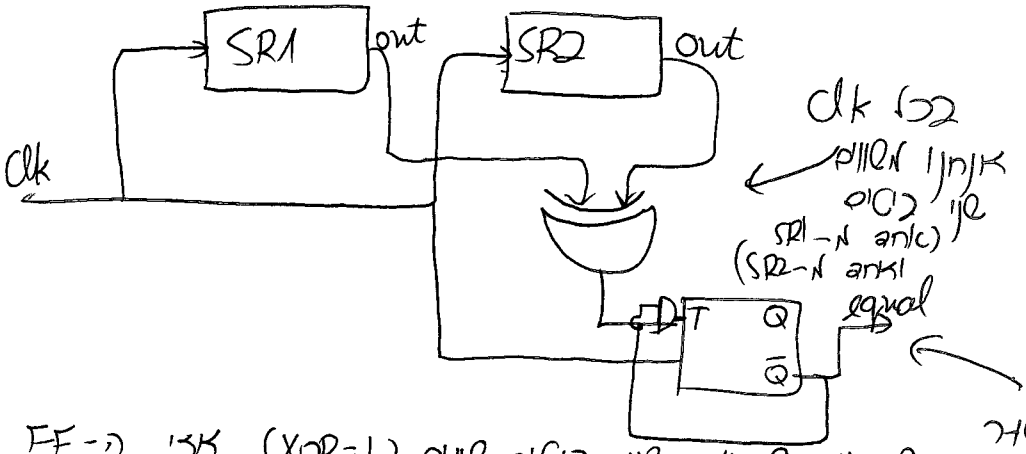
פתרון:



המונה שלנו הוא
נשים ♥ ל- X_3 מתחברים בכל 8 פעימות שעון
ולכן X_5 X_4 X_3 מתפרש כמחזוריות.



תרגיל: לחבר 2 יח' SR 4-bit המכילות שני מס'
קבל מעגל שמשווה את שני המספרים.



מהנתון שני מספרים:
1010
0110
1100
0000
Q

0-בתים
1-שונים

ה-FF בא
אחרי רגע הוא
היה ביט כשהו
שונה בכלל. שר
ה-AND פונה

אינו לא שמר
המצב-שהיה
שני מספרים.
לכן נעלם
כמה.

שבלעד שמידע שני ביטים שונים (XOR=1) אזי ה-FF
ושאר 0 תמיד.

תרגיל: נמונים 2 מונים:

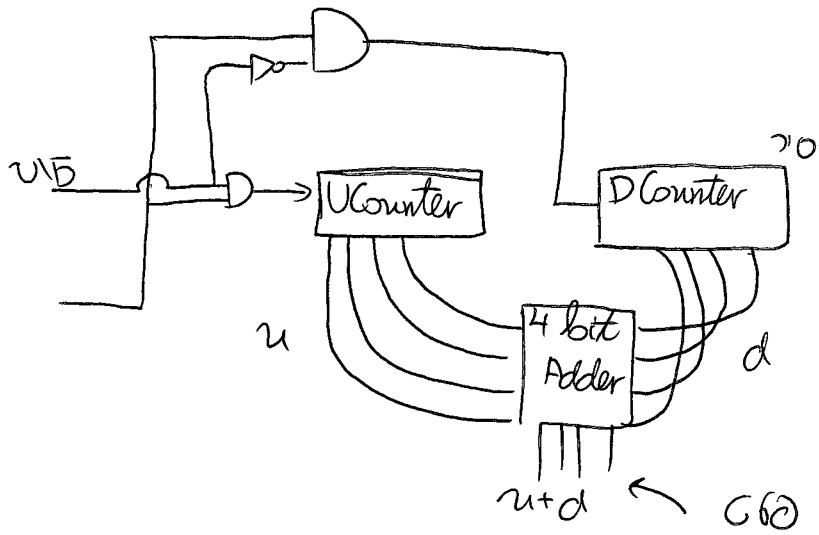
4 bit UCounter (סופר וחוזר)

4 bit DCounter (סופר ומחסיר)

בנה UDCounter

U\D כניסה קו שאומר 0 - עולה

1 - יורד.



נשמור שני מונים.

כל פעם שנרצה להחסיר

נוריד מ - DCounter

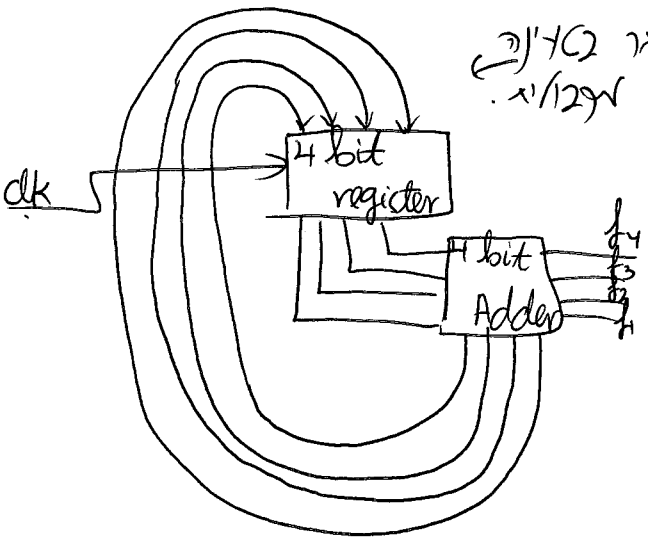
ונוסיף ל - UCounter

הפלט יהיה הסכום

של שני המונים.

תרגיל: תכנן מונה U\D שיכול להתקדם בקפיצות

של 1±, 2±.



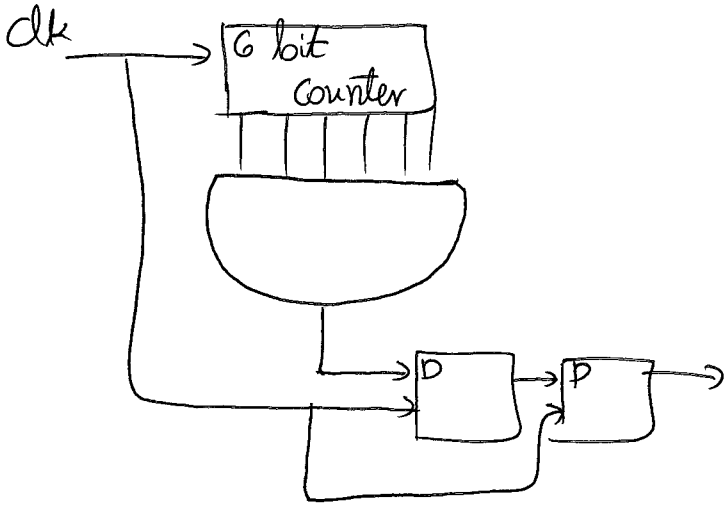
$f_1 = \bar{x}$

$f_2 = \bar{x}$

$f_3 = \bar{x} + y$

$f_4 = \bar{y}$

מכיוון בכל עדכון שאנו יוצא "1" אחרי 65
מתקבלים שעון.
(עון- counter 6 mod



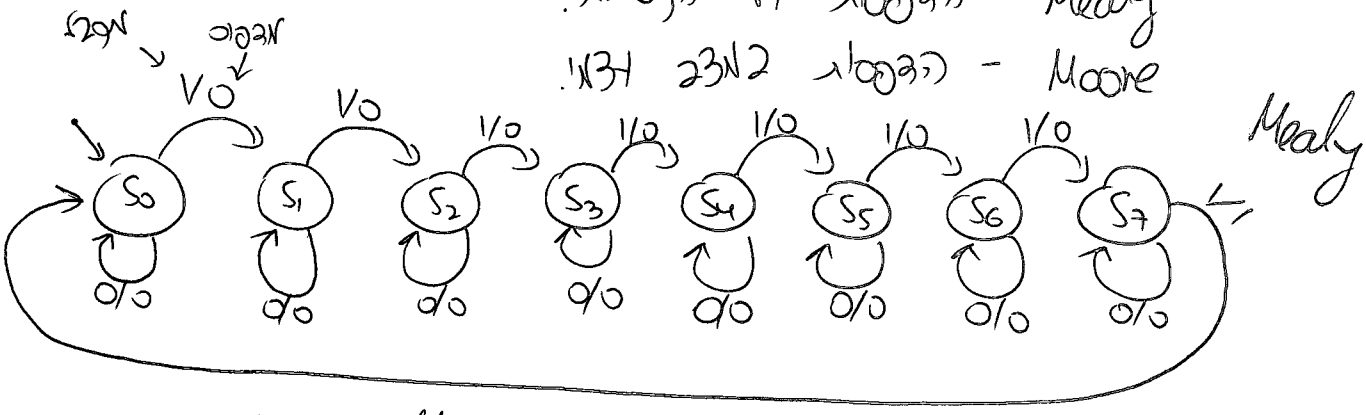
26.12.2005

מבוא מחשבים - ארז'ן

תכנן מעגל שמפיק 0 "1" שמין שמגיע.

בדוגמא: כלומר: 0110111010101111
00000000000000100

Mealy - הפלט תלוי בכניסה.
Moore - הפלט במצב הבא.



(*) מצב משגיח רק כשמשגיח ה- clk הפלט יוצא כצא מ"ב.

מימוש מכונת המצבים ע"י מערגל

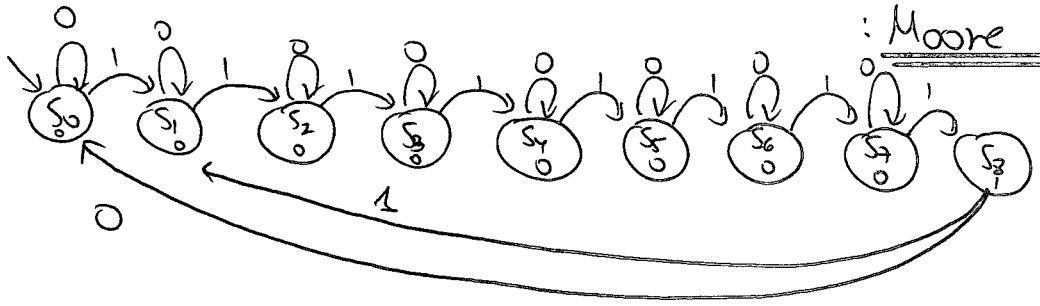
Present State			Next State		Z		$T_1T_2T_3$	
Q_1	Q_2	Q_3	X=0	X=1	X=0	X=1	X=0	X=1
0	0	0	000	001	0	0	000	001
0	0	1	001	010	0	0	000	011
0	1	0	010	011	0	0	000	001
0	1	1	011	100	0	0	000	111
1	0	0	100	101	0	0	000	001
1	0	1	101	110	0	0	000	011
1	1	0	110	111	0	0	000	001
1	1	1	111	000	0	1	000	111

$T_1(x,Q_1,Q_2,Q_3)=\sum(11,15)=xQ_2Q_3$

$T_2(x,Q_1,Q_2,Q_3)=\sum(9,11,13,15)=xQ_3$

$T_3=x$

מכונת Moore:

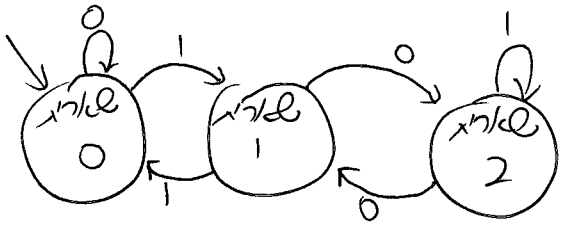


(*) תכנן מדרג שמוציא "1" אם הגיע שמתקבל MSB first מתחלף 3-2 בשורות 1.

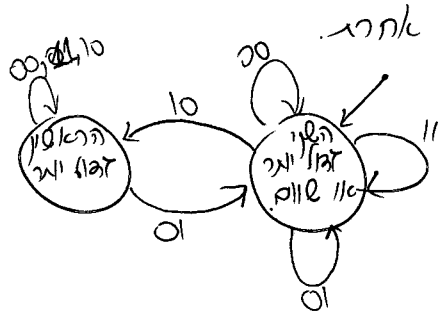
בדיקה: קלט: 1010

פלט: 1001

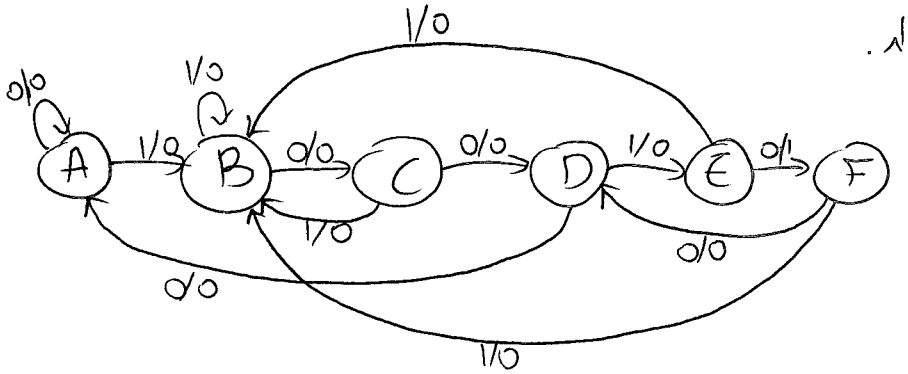
מכונת MOORE:



(*) מדרגים שני מס' בעבודת LSB first. תכנן מדרג שמוציא "1" אם הראשון גדול יותר, "0" אחרת.



(*) תכנן מדרג שמזהה את הסדרה 01001 בקלט סדרתי עם חפיפה.



שיטה עבור מס' המצבים:

1. רושמים את המצבים באין סופיים. סופיים משמאלם - יש סיכוי שמובטח באותו מצב.
2. הפרד את כל המצבים שונים בפלט שלהם.
3. הפרד את כל המצבים שיכולים עתיד שאין ביב שיני. עדיין בצורה אחה לכל שונות.

PS	NS X=0	NS X=1
A	E/0	D/1
B	F/0	D/0
C	E/0	B/1
D	F/0	B/0
E	C/0	F/1
F	B/0	C/0

⇓

(ABCDEF)
↓
(ACE)(BDF)
↓
(ACE)(BD)(F)
↓
(AC)(E)(BD)(F)
↓
אז נמן עבתקדם ימה.

נפעיל את השיטה על האוטומט הקודם:

PS	NS X=0	NS X=1
A	A/0	B/0
B	C/0	B/0
C	D/0	B/0
D	A/0	E/0
E	F/1	B/0
F	D/0	B/0

(ABCDEF)
↓
(ABCDF)(E)
↓
(ABCF)(D)(E)
↓
(AB)(CF)(D)(E)
↓
(A)(B)(CF)(D)(E)
↓
אז נמן עבאמצע.

לא עברנו את האוטומט נמחק את F עבורנו ולכן מה שנכנס אליו נשאר אל C. את הקשת היוצאת משם נמחק. הן אינן רלוונטי ב-C.

2.1.2006

מבוא מחשבים - ארז'ן

דברים באוסמבלר:

(*) שפת מכונה MIPS.

(*) כל מילה מאותחלת 2-4 בתים (Bytes)

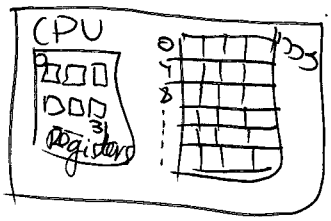
(*) add $t1, $a1, $a2

משמעו הרגיסטר $t1 יקבל את הערכים $a1+$a2

(*) sw $t0, 0($t1)

כאן אורך הרגיסטר $t0 את הערך מהזיכרון שכתובתו ב- $t1. המס' אפני הסוגריים הוא ה- offset

כלומר את הערך בכל 2-0 בתים נוספים.



כשיש סוגרים הוא הלוך וחזרה וכשאין סוגרים רגיסטר.

(*) sw $t2, 0($t1)

שומר את הערך שבזיכרון בכתובת 0($t1) רגיסטר $t2.

(*) jr $ra

ברגע שמגיעים אפשרי כל כלל הפונקציה הנוכחית הסתיימה והקוראת יתקדם לשלב שמגיע הוא רגר בפרמטרים הקוראת. (הוא נקרא בדגם אפשרי הבאה)

(*) move $s0 $zero שפה ר $zero ← $s0

(*) slt $t0 $s0 $s3 שפה ר:

if $s0 < $s3

then $t0 = 1

slt = set if less than

(*) פקודת beq: beq $t0, $zero, exit1

בודק האם $t1 = $zero אם כן לוקח ל- exit1.

הפרוצ' sort מממש מערך ע"י אינדקס ה-C הבאה:

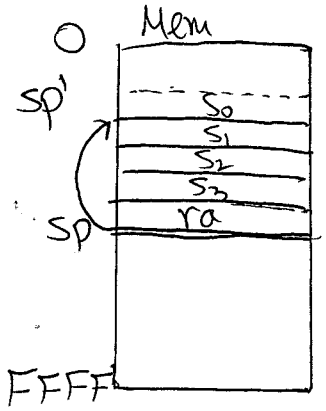
```
Sort(int V[], int n)
{
  int i,j;

  for (i=0; i<n; i=i+1)
    for(j=i-1; j≥0 && V[j]>V[j+1]; j=j-1)
      swap(V,j);
}
```

(*) bne כמו ה- beq רק not.

(*) jal - jump and link

בהפעלת swap jal האסמבלר יקפוץ לתחילת הפרוצדורה swap ובנוסף ישמור את הכתובת הבאה בוא כ- $ra.

(*) רגיסטרים שמתחילים ב- $s הם מאותחלים ערכים בין פרוצדורות. אבל אסור לשנות ערכם. (מותר לשנות אבל אז להחזיר)



(*) קונבנציה:

אסור לפנות מקום מעבר לגבולות שמותר (הגבול מעל ל-sp)

sp = stack pointer

(*) zoot.tau.ac.il מחשב שמריץ MIPS

להיכנס ע"י nova כרגיל ואז להקיש

ssh username@zoot.tau.ac.il

(*) בקומפיילר מגדיר מס' רגיסטר לפי #to

($t0 = $8)

gcc main.o example.o יקמפל.

objdump -d example.o - מציג פירוק אסמבלר

gdb - ג'בגאר

עקרון הפרוצדורה:

swap(int v[], int k)

a_0 = v, a_1 = k

יחליף את v[k], v[k+1]

כל מה שיש אפליו $ הוא כתיבה =

Procedure body

```
swap: add   $t1, $a1, $a1      # reg $t1 = k * 2
      add   $t1, $t1, $t1      # reg $t1 = k * 4
      add   $t1, $a0, $t1      # reg $t1 = v + (k * 4)
                               # reg $t1 has the address of v[k]
      lw    $t0, 0($t1)        # reg $t0 (temp) = v[k]
      lw    $t2, 4($t1)        # reg $t2 = v[k + 1]
                               # refers to next element of v
      sw    $t2, 0($t1)        # v[k] = reg $t2
      sw    $t0, 4($t1)        # v[k+1] = reg $t0 (temp)
```

Procedure return

```
      jr    $ra                # return to calling routine
```

אמיין מערך בגודל n.

sort(int v[], int n)

a_0 = v, a_1 = n

Saving registers

```
sort: addi  $sp,$sp, -20       # make room on stack for 5 registers
      sw    $ra, 16($sp)       # save $ra on stack
      sw    $s3,12($sp)        # save $s3 on stack
      sw    $s2, 8($sp)        # save $s2 on stack
      sw    $s1, 4($sp)        # save $s1 on stack
      sw    $s0, 0($sp)        # save $s0 on stack
```

Procedure body

```
Move parameters
      move  $s2, $a0           # copy parameter $a0 into $s2 (save $a0)
      move  $s3, $a1           # copy parameter $a1 into $s3 (save $a1)

Outer loop
      move  $s0, $zero         # i = 0
for1tst:slt $t0, $s0, $s3      #  reg $t0 = 0 if $s0 ≥ $s3  (i ≥ n)
      beq   $t0, $zero, exit1  # go to exit1 if $s0 ≥ $s3  (i ≥ n)

Inner loop
      addi  $s1, $s0, -1       # j = i - 1
for2tst:slti $t0, $s1, 0       # reg $t0 = 1 if $s1 < 0 (j < 0)
      bne   $t0, $zero, exit2  # go to exit2 if $s1 < 0 (j < 0)
      add   $t1, $s1, $s1      # reg $t1 = j * 2
      add   $t1, $t1, $t1      # reg $t1 = j * 4
      add   $t2, $s2, $t1      # reg $t2 = v + (j * 4)
      lw    $t3, 0($t2)        # reg $t3  = v[j]
      lw    $t4, 4($t2)        # reg $t4  = v[j + 1]
      slt   $t0, $t4, $t3      # reg $t0 = 0 if $t4 ≥ $t3
      beq   $t0, $zero, exit2  # go to exit2 if $t4 ≥ $t3

Pass parameters and call
      move  $a0, $s2           # 1st parameter of swap is v (old $a0)
      move  $a1, $s1           # 2nd parameter of swap is j
      jal   swap               # swap code shown in Figure 3.24

Inner loop
      addi  $s1, $s1, -1       # j = j - 1
      j     for2tst            # jump to test of inner loop

Outer loop
exit2: addi $s0, $s0, 1        # i = i + 1
      j     for1tst            # jump to test of outer loop
```

Restoring registers

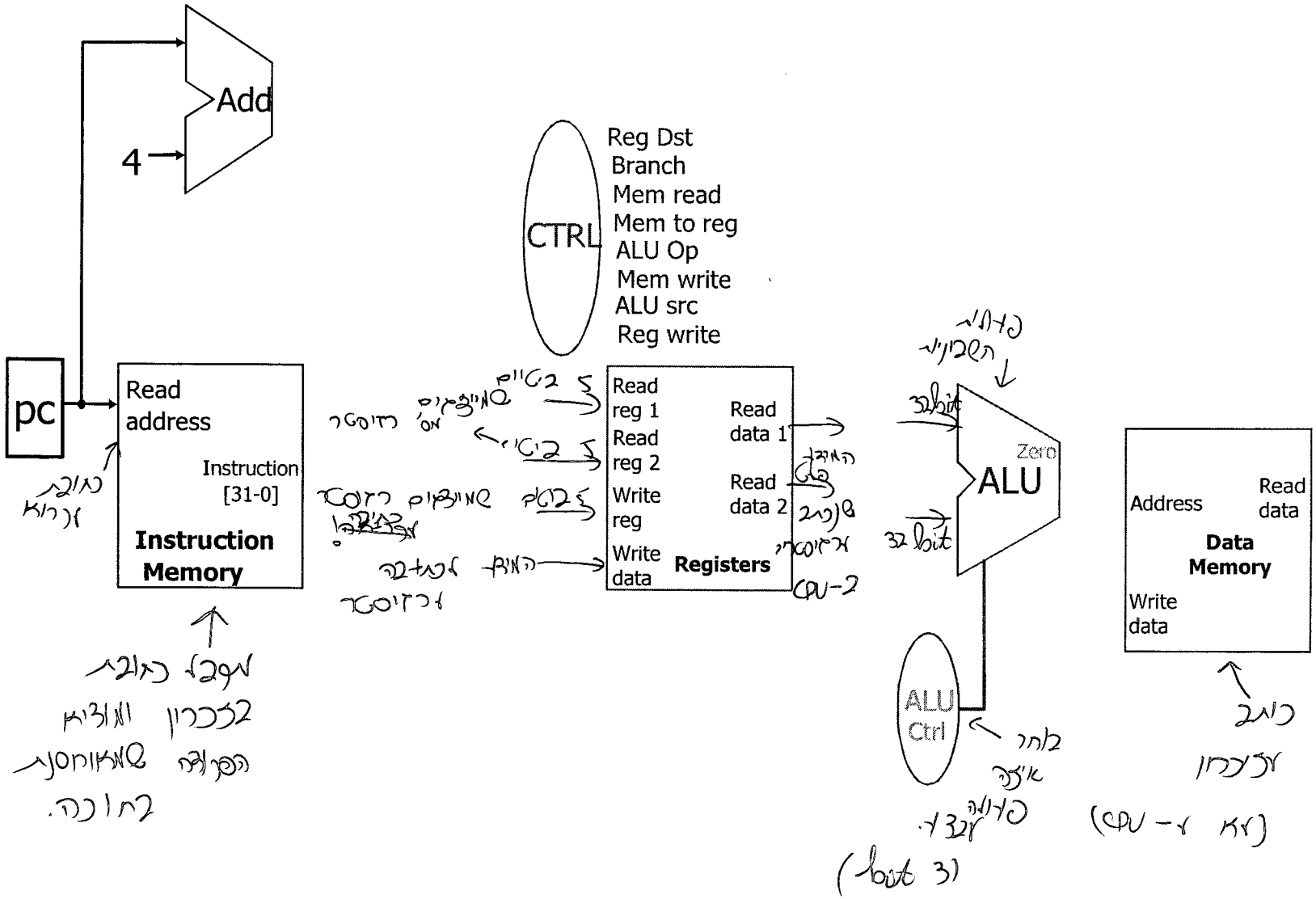
```
exit1: lw   $s0, 0($sp)        # restore $s0 from stack
       lw   $s1, 4($sp)        # restore $s1 from stack
       lw   $s2, 8($sp)        # restore $s2 from stack
       lw   $s3,12($sp)        # restore $s3 from stack
       lw   $ra,16($sp)        # restore $ra from stack
       addi $sp,$sp, 20        # restore stack pointer
```

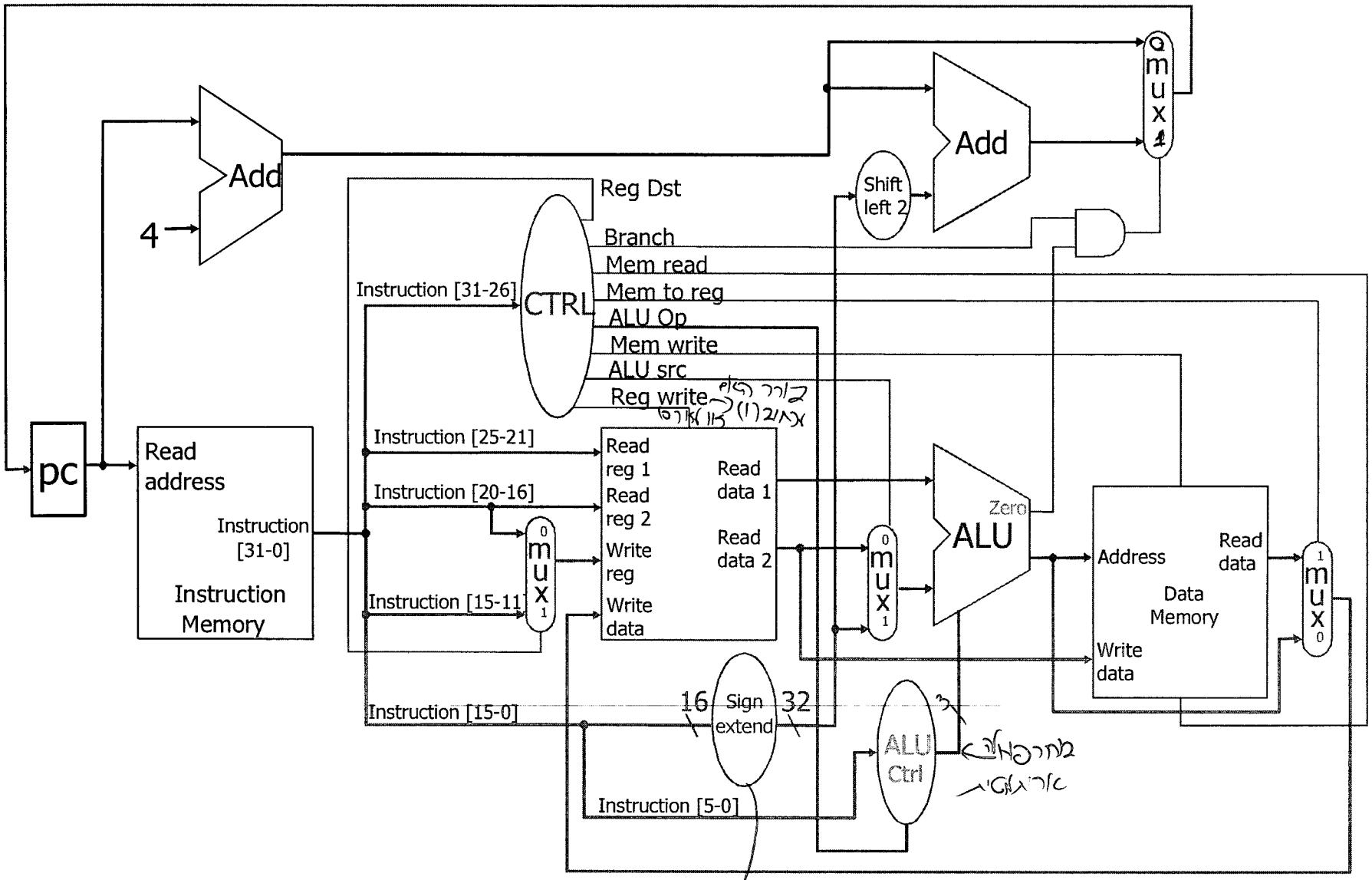
Procedure return

```
       jr   $ra                # return to calling routine
```

רשימת רגיסטרים ב- CPU.

Register Name	Number	Usage
zero	0	Constant 0
at	1	Reserved for assembler
v0	2	Expression evaluation and
v1	3	results of a function
a0	4	Argument 1
a1	5	Argument 2
a2	6	Argument 3
a3	7	Argument 4
t0	8	Temporary (not preserved across call)
t1	9	Temporary (not preserved across call)
t2	10	Temporary (not preserved across call)
t3	11	Temporary (not preserved across call)
t4	12	Temporary (not preserved across call)
t5	13	Temporary (not preserved across call)
t6	14	Temporary (not preserved across call)
t7	15	Temporary (not preserved across call)
s0	16	Saved temporary (preserved across call)
s1	17	Saved temporary (preserved across call)
s2	18	Saved temporary (preserved across call)
s3	19	Saved temporary (preserved across call)
s4	20	Saved temporary (preserved across call)
s5	21	Saved temporary (preserved across call)
s6	22	Saved temporary (preserved across call)
s7	23	Saved temporary (preserved across call)
t8	24	Temporary (not preserved across call)
t9	25	Temporary (not preserved across call)
k0	26	Reserved for OS kernel
k1	27	Reserved for OS kernel
gp	28	Pointer to global area
sp	29	Stack pointer
fp	30	Frame pointer
ra	31	Return address (used by function call)


Add
4
pc
Read address
Instruction [31-0]
Instruction Memory
CTRL
Reg Dst
Branch
Mem read
Mem to reg
ALU Op
Mem write
ALU src
Reg write
Read reg 1
Read reg 2
Write reg
Write data
Read data 1
Read data 2
Registers
32bit
32 bit
Zero
ALU
ALU Ctrl
Address
Read data
Data Memory
Write data



פונקציה אמת אפיין
16 ביטים ל-32
ע"י הוספת 1-ים מימין המספר שמייצג "פלוס" אם-מספר
1-ים אם חיובי

3

Field	0	rs	rt	rd	shamt	funct
Bit positions	31–26	25–21	20–16	15–11	10–6	5–0

a. R-type instruction → sub, or, and

Field	35 or 43	rs	rt	address
Bit positions	31–26	25–21	20–16	15–0

b. Load or store instruction

Field	4	rs	rt	address
Bit positions	31–26	25–21	20–16	15–0

c. Branch instruction

Addition immediate (with overflow)

אותה מס' (2|1)
אריתמטי.

`addi rt, rs, imm`

8	rs	rt	imm
6	5	5	16

opcode

Jump

`j target`

2	target
6	26

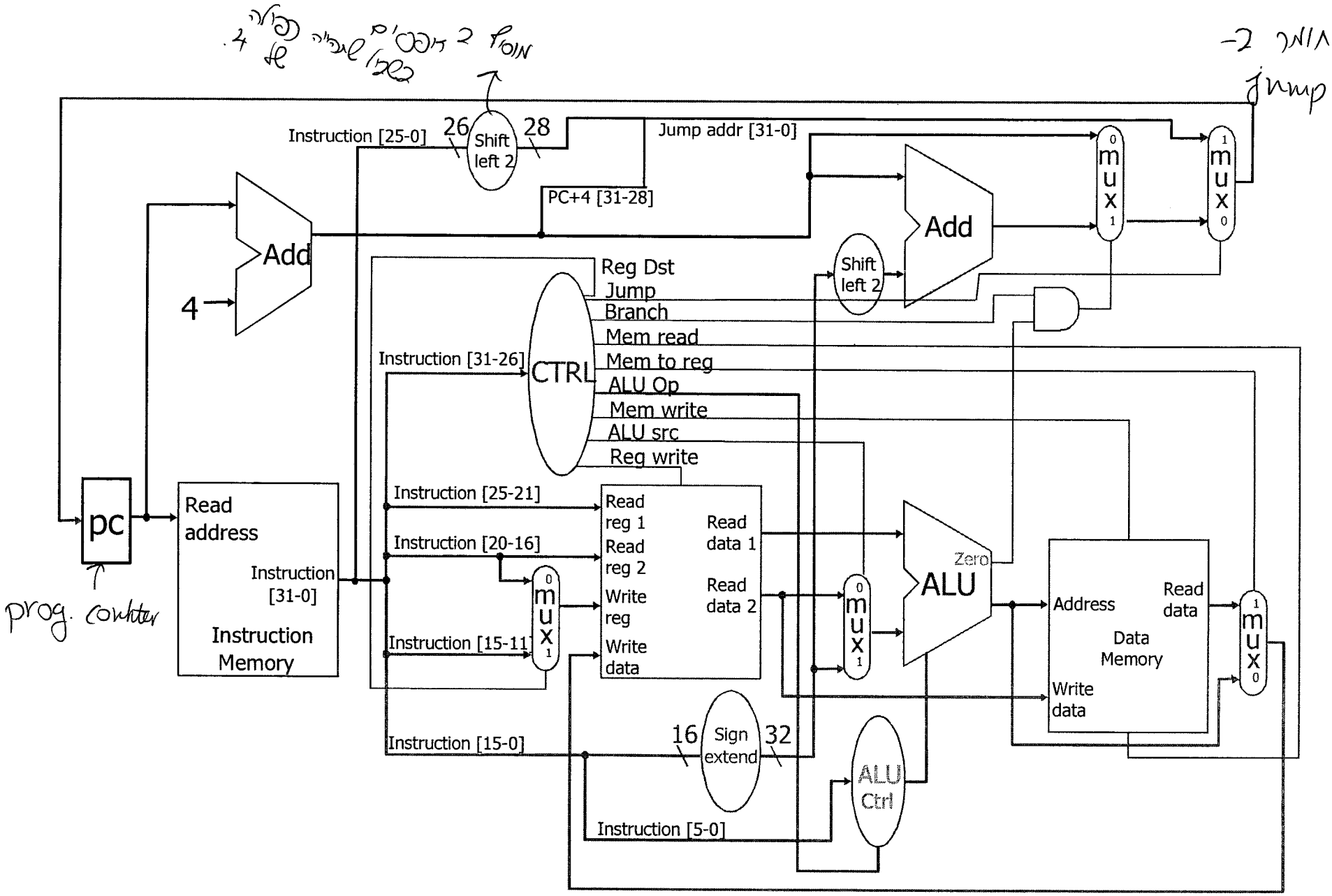
Unconditionally jump to the instruction at target.

Jump and link

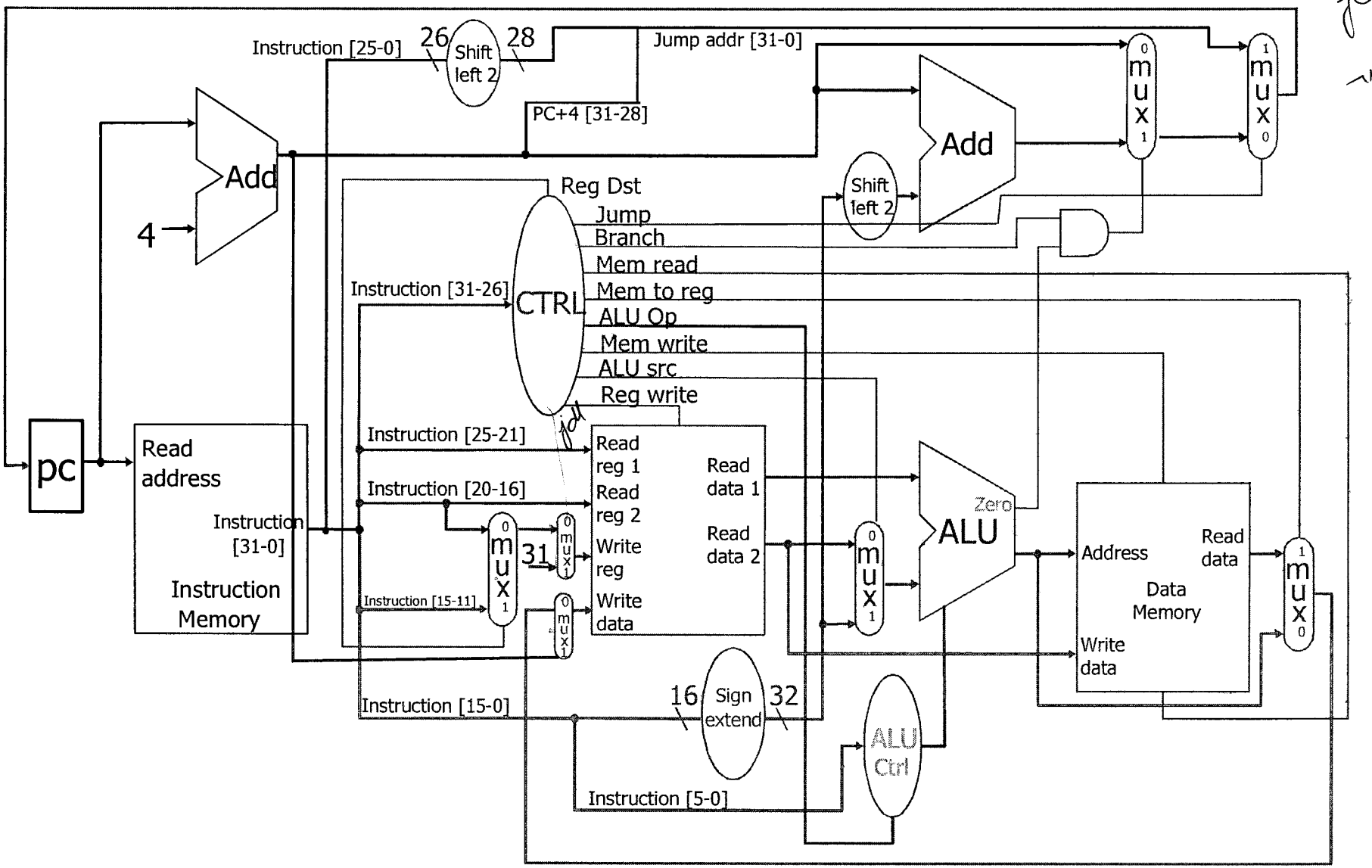
`jal target`

3	target
6	26

Unconditionally jump to the instruction at target. Save the address of the next instruction in register `$ra`.


מוסף 2 דיפס'ים בשביל שביה כפיר של 4.
אומר 2-
jump
Instruction [25-0]
26
Shift left 2
28
Jump addr [31-0]
PC+4 [31-28]
Add
4
Add
Shift left 2
mux 1
mux 0
Reg Dst
Jump
Branch
Mem read
Mem to reg
ALU Op
Mem write
ALU src
Reg write
CTRL
Instruction [31-26]
pc
prog. counter
Read address
Instruction [31-0]
Instruction Memory
Instruction [25-21]
Instruction [20-16]
Instruction [15-11]
Instruction [15-0]
Instruction [5-0]
Read reg 1
Read reg 2
Write reg
Write data
Read data 1
Read data 2
16
Sign extend
32
ALU
Zero
ALU Ctrl
Address
Read data
Data Memory
Write data

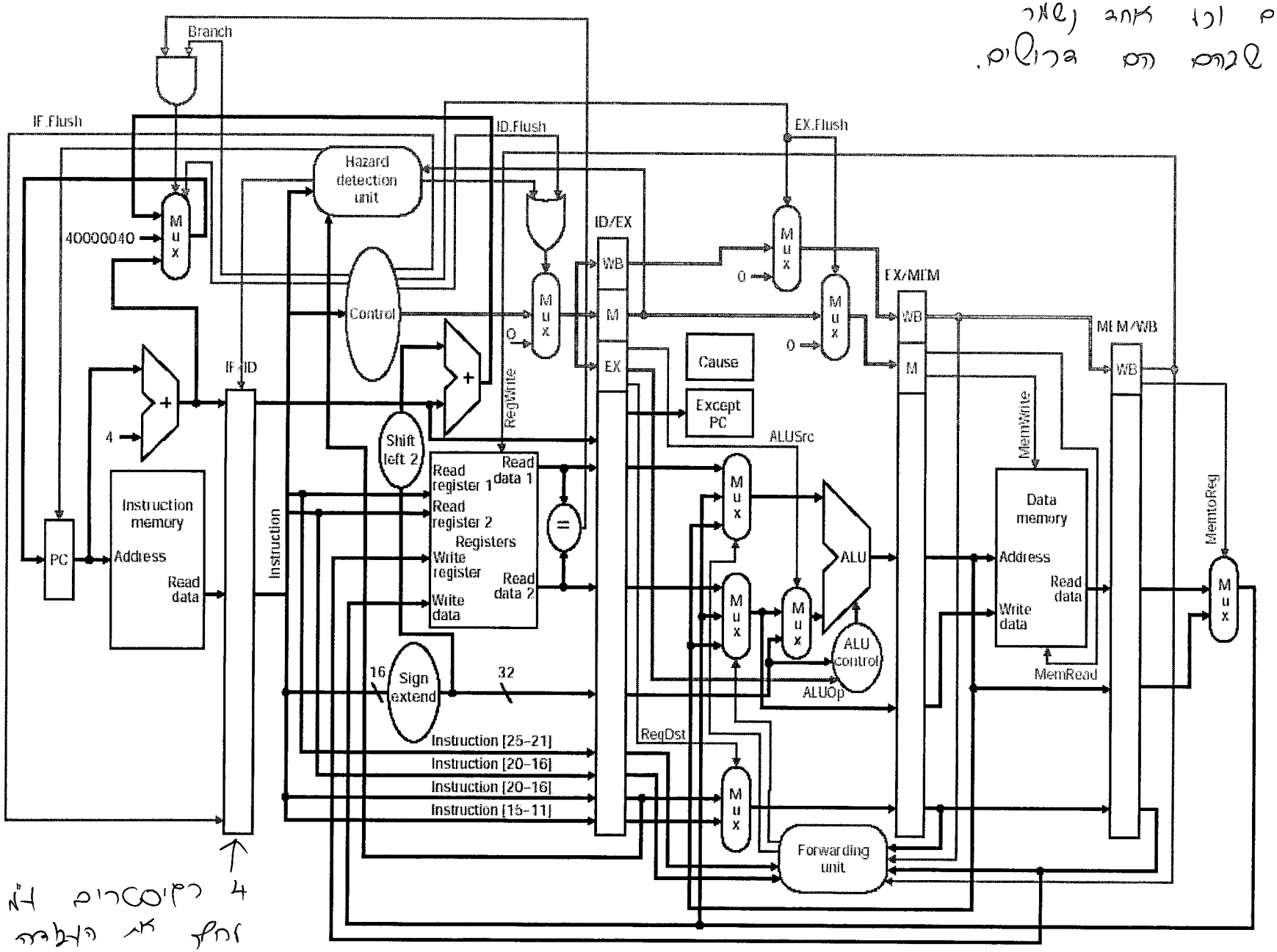
5


pc
Add
4
Read address
Instruction [31-0]
Instruction Memory
Instruction [25-0]
26
Shift left 2
28
Jump addr [31-0]
PC+4 [31-28]
Add
Shift left 2
mux 0
mux 1
Reg Dst
Jump
Branch
Mem read
Mem to reg
ALU Op
Mem write
ALU src
Reg write
CTRL
Instruction [31-26]
Instruction [25-21]
Instruction [20-16]
Instruction [15-11]
Instruction [15-0]
Instruction [5-0]
Read reg 1
Read reg 2
Write reg
Write data
Read data 1
Read data 2
31
16
Sign extend
32
ALU
Zero
ALU Ctrl
Address
Read data
Data Memory
Write data

jal N/A
-2
jump

שים ♡ : רוי' הבדיה
מתבלבלים לגו אחה נשאר
ושאר שבמה הם ברולים.

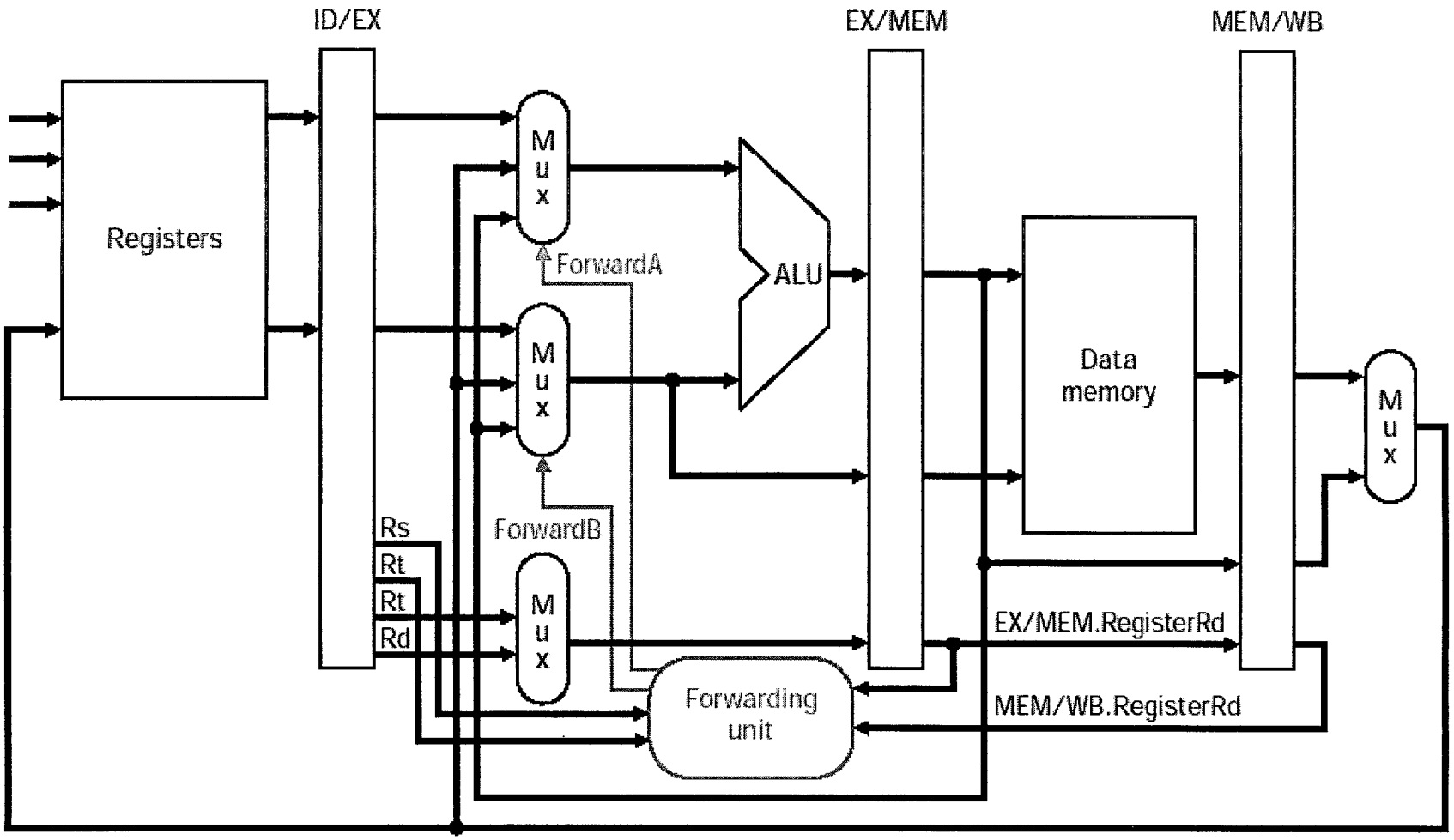
מבנה המעבד שלנו



4 כי מסרים ל"מ
נחלף את הערכים ל-5
שלבים בלעבדו
pipeline.

16.1.06

1


ID/EX
EX/MEM
MEM/WB
Registers
M u x
M u x
M u x
ALU
Data memory
M u x
ForwardA
ForwardB
Rs
Rt
Rt
Rd
EX/MEM.RegisterRd
MEM/WB.RegisterRd
Forwarding unit

3

Hazards

האפס ושאר אפס
יכו את כתבו $6 $5 $0 add
אסור אין אל Forwarding

EX hazard	If (EX/MEM.RegWrite and (EX/MEM.RegisterRd ≠ 0) and (EX/MEM.RegisterRd = ID/EX.RegisterRs)) then ForwardA = 10 If (EX/MEM.RegWrite and (EX/MEM.RegisterRd ≠ 0) and (EX/MEM.RegisterRd = ID/EX.RegisterRt)) then ForwardB = 10
MEM hazard	If (MEM/WB.RegWrite and (MEM/WB.RegisterRd ≠ 0) and (MEM/WB.RegisterRd = ID/EX.RegisterRs)) and (EX/MEM.RegisterRd ≠ ID/EX.RegisterRs)) then ForwardA = 01 If (MEM/WB.RegWrite and (MEM/WB.RegisterRd ≠ 0) and (MEM/WB.RegisterRd = ID/EX.RegisterRt)) and (EX/MEM.RegisterRd ≠ ID/EX.RegisterRt)) then ForwardB = 01
Load hazard	If (ID/EX.MemRead and ((ID/EX.RegisterRt = IF/ID.RegisterRs) or (ID/EX.RegisterRt = IF/ID.RegisterRt))) then *stall the pipeline*
Branch hazard	If ((IF/ID.opcode = beq) and (FastComparator = 1)) then *stall the pipeline*

אם יש
בעיה
כי אין מי
את הדרך
מהכרון.

אם יש
בעיה branch

ואם בדרך יהיה
לו הבעיה הבאה
אל שמים nop
ומאפסים את הרגיסטר
ב- IF\ID

16.1.06

מבנה מחשבים - ארז'ו

Pipeline - רעיון כמו השרשרת

הוראה מתחלקת ל-5 שלבים

Information Fetch - IF

Info. Decode - ID

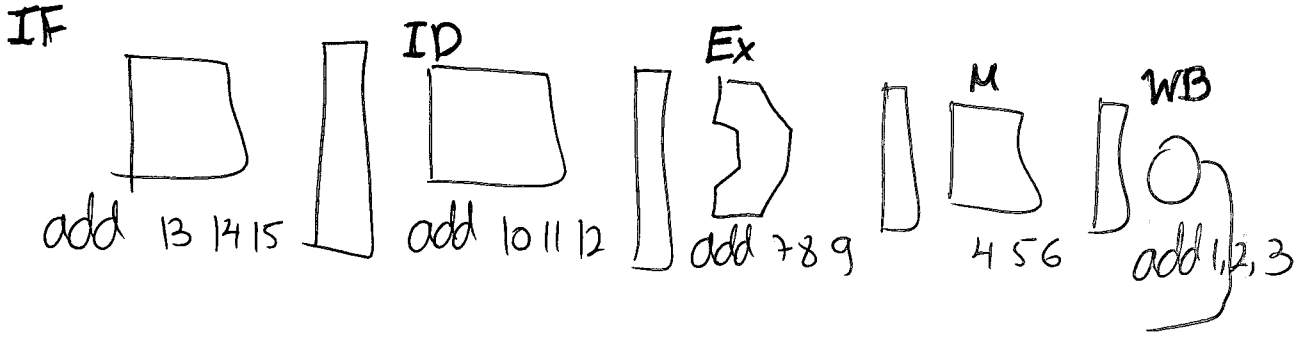
Execute - EX

Memory - M

Write Back - WB

נסתכל על הדוגמא:

```
add $1 $2 $3
add $4 $5 $6
add $7 $8 $9
add $10 $11 $12
add $13 $14 $15
```



מה קורה ב- Cycle הראשון? אין רגיסטרים נכתבים ונקראים
1 ↓ 11,12 ↓

הגוכנות במבנה הנוכחי רצה "יחלק" בלי בעיה.
אולם אם היינו משנים את סדר 3 ההוראות

```
add $7 $8 $4
```

בעיה כן היא מסוג Data Hazard

מה שהיינו עושים כדי "לדחוף" את התשובה מ- Ex
שהתבצע כבר אחשה (הראה סרטון אחוב רופס)
הוא מה שמתרחש אלף הוא Fowording

אם כן, ה- MUX הדליון (בשרטוט אחוב ב)
הוא אפס 00 = FowardA ! FowardB=10
כדי שיתובר הכניסה המתאימה.

(*) אם הפעולה הייתה add $7 $8 $1 התשובה של
1 נמצאת ב שצבים הראה (לא שלב יחוב)
היינו עושים FowardA=00 ; FowardB=01
בכללי: Foward =00 אין לקפוץ
Foward =01 לקפוץ המוצאה מ- MWB
Foward =10 לקפוץ המוצאה מ- EX/MEM
(FowardA / FowardB רבלו אחה אכנים את הדרך
אלינה כניסה)

23.01.2006

מבנה מחשבים - ארז'ן

Delayed Branch Slot

```
loop: lw $2, 100($3)
      addi $3, $3, 4
      beq $3, $4, loop
```

בדוגמא הזו אחרי ה- beq כן פנים יבצע- nop.
ניתן לשנות את סדר הפקודות בשביל לחסוך את ה- nop.
בתנאי שזה לא יגרום להתנגשות

```
loop: addi $3, $3, 4
      beq $3, $4, loop
      lw $2, 96($3)
```

ניתן בדיוק Delayed Branch
שים ♥ - כבר ירא מחדש הקומפיילר עדיין משתמש
בדיה אלא אם כן אמר אחרת.

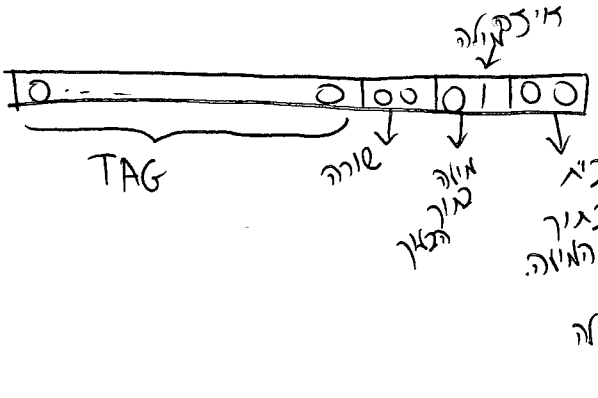
Cache

Cache בגודל 16 מילים.
באך ל- Cache יכול 4 מילים.
מיפוי בשיטת Direct Mapped
סדרת גישות זיכרון בלבד: 1,4,8,5,20,17,19,56,9,11
M M M H M M H M M H

	Valid	tag	data			
			מילה 0	מילה 1	מילה 2	מילה 3
(0)						
(1)						
(2)						
(3)						

Valid - מילה מאותחלת
יהיו אפס. כשמוכנס
ערך כזה הופך ל-1.

כתובת בבתים:



כל עוד בבלוק הזה, אם הקבלו עליך אז הוא עליך בבתים

הרעיון ב"שאילה" הינו כך שבפעם את הזיכרון של השורה כשאין כאם בדרך עדיין כשיש משהו אבל כך את הדרך המתאים יש בדרך עדיין (M= Miss) (ואחרי דרך מהכרחי) אבל כשיש את הדרך המתאים (מופיעים גם ה- tag) אז פגיעה (H=hit)

בגלל שה- Direct Map "פגיעה" רק ב-2 בתים.
שיטה נוספת עיפול ב- Cache היא LRU
ושם את מוחקים דרך מי שמחשיב אלא מי מוחקים את הדרך הלא מחרון שהשתמשו בו.

	כמן	את כמן	בלוק 0	בלוק 1	בלוק 2	בלוק 3
M	1	1	0-3 (1)			
M	2	4	0-3 (1)	4-7 (2)		
M	3	8	0-3 (1)	4-7 (2)	8-11 (3)	
H	4	5	0-3 (1)	4-7 (4)	8-11 (3)	
M	5	20	0-3 (1)	4-7 (4)	8-11 (3)	20-23 (5)
M	6	17	16-19 (6)	4-7 (4)	8-11 (3)	20-23 (5)
H	7	19	16-19 (7)	4-7 (4)	8-11 (3)	20-23 (5)
M	8	56	16-19 (7)	4-7 (4)	56-59 (8)	20-23 (5)
M	9	9	16-19 (7)	8-11 (9)	56-59 (8)	20-23 (5)
H	10	11		8-11 (10)		

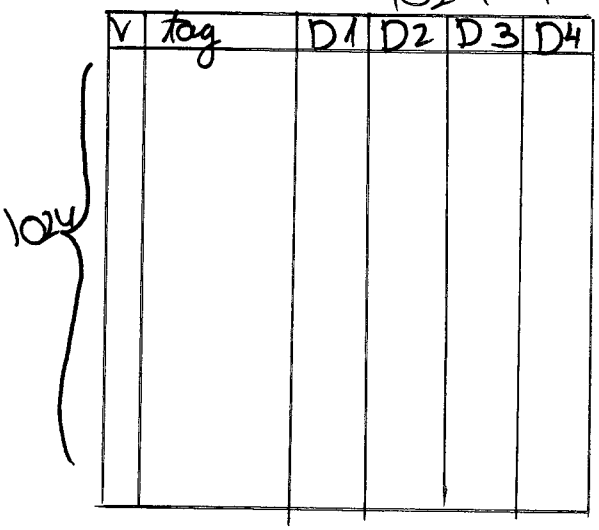
כי פעם שבאנו אור המס בסמ' ישנו כך נשאר בדיסק אומר Hit
(אם הבאר נשאר אנו בלוק 0)

ארכיטקטורה זו Cache יש 1024 בלוקים בגודל 4 מילים. חלק מה קיבלו ובדוגמא.

1. Direct Mapped

גודל: $1024 \cdot 4 \cdot 4 = 16KB = 128Kbit$

(בלוקים, מילים, בתים)



בדוגמא:

$1024(1+18+4\cdot 32) = 147Kbit$

valid, tag, words

tag = 18

נקבלים

כי יש 10 בתים בזיכרון

כי יש 1024 בלוקים.

2. 4-Way Set Associative

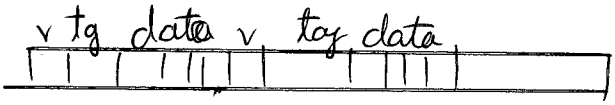
גודל- נשאר אותו דבר 16KB

בדוגמא- $4\cdot 256(1+20+4\cdot 32) = 149Kbit$

valid, tag, 4×word

3. Fully Associative

(בהנחה שמעתיקים בצורה רגילה ולא LRU)



כי הוספנו ב"שורה" אחת

tag=28 כי אין שורות.

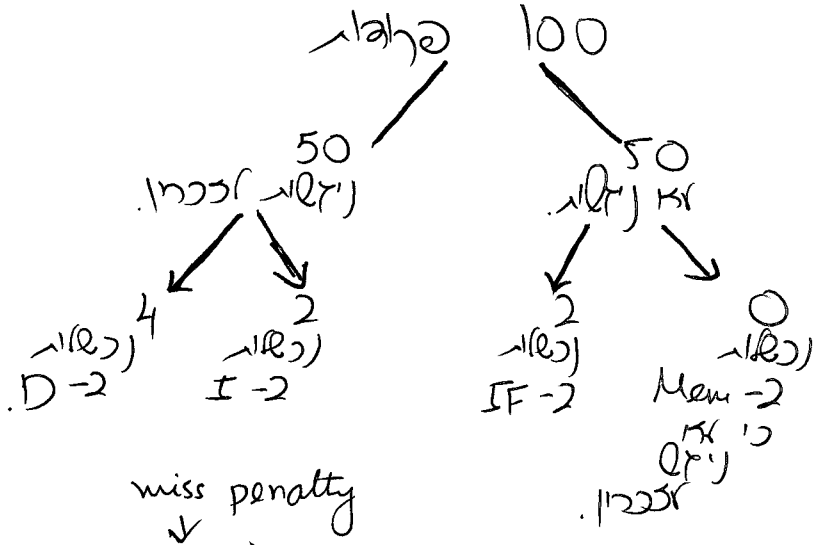
בדוגמא $1024\cdot(1+28+128) = 157Kbit$ ~~256~~

תרגיל ל 3 מעלות לפי מרכיבי ארכיטקטורה Cache שונה

	A	B	C
זמן בסיס	1	4	4
D-miss rate ← Data	8	5	4
I-miss rate ↓ Instruction Fetch	4	2	2

miss penalty = 6 + זמן בסיס

כמה cycles יוקחו סך הכל בממוצע במכונה A.
בהינתן ש 1/2 מהסיכויים הם שגיאות! שגיאות.



miss penalty
8·(1+6)

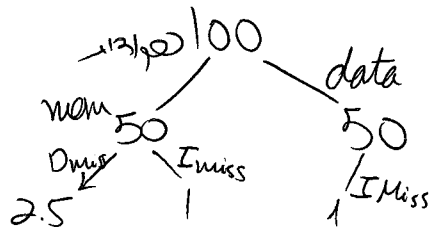
סך הכל בסך הכל 8 פעולות כאמור,

30.1.2006

מבנה מחשבים - ארז'ל

המשך... (שיעור לפני)

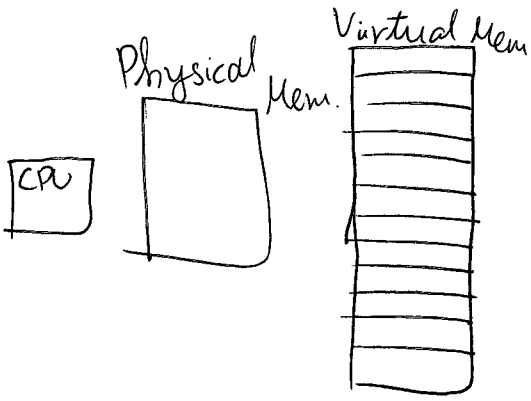
נחשב CPI של מכונה B:



miss penalty = ~~100~~ 6 + 4 = 10

$$CPI_B = \frac{145 + 44}{100} = 1.89$$

בשיעור הבא נראה נחשב CPI_C.



Virtual Memory

Virtual Mem

כרגע "קצת" ארוכות בצד.

מחולק "דפים"

ישנם 3 מצבים:

- invalid
- physical
- disk

מעכשיו נחשוב איפה נמצא המידע

אז נמצא כרגע (invalid)

בזיכרון הפ'ס' (physical)

או בדיסק (disk)

כי פרט שנשלח "מחוץ" לעולם המידע ב- VM ירחבן.

תרגיל 8

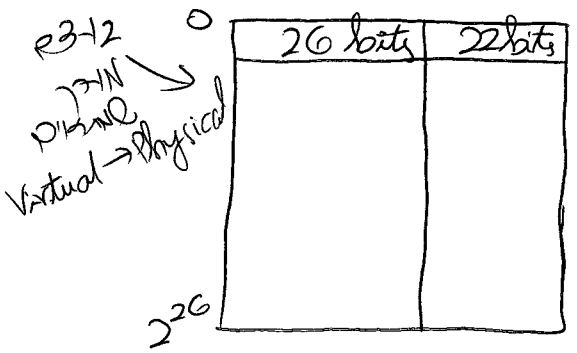
כתובת וירטואלית - 40 ביטים.
גודל דף - 16KB
כתובת פיזית: 36 ביטים.
מה גודל טבלת המיפוי?

virtual [40 bits]
physical [36 bits]

בדומה לסעיף א' הכתובת הווירטואלית ונקבה אבל
בגינה page מפורבך? אין, לסעיף א' ה-26 ביטים
הראשונים. כי 16 הביטים התחתונים הם "באורך הדף"
(כי $2^{14}=16KB$ שזה גודל ה-page)
לפי טבלת המיפוי מוצאים את 22 הביטים שמהווים
את הכתובת הפיזית. הגודל הוא תמיד

כתובת ← [22 bits | 14 bits]

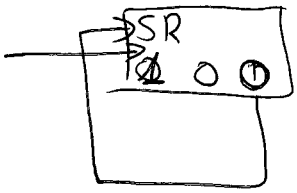
גודל הטבלה



$2^{26} \times (22+26)$

pagefault - כל המידע המבורש נמצא בדיסק, ולא
עכשיו אותו. חלק מהמידע מאוחר

תרגיל: בנה רכיב A שקולט "1"
ומשך מחזור שלו אחד כל 3 מחזורים.

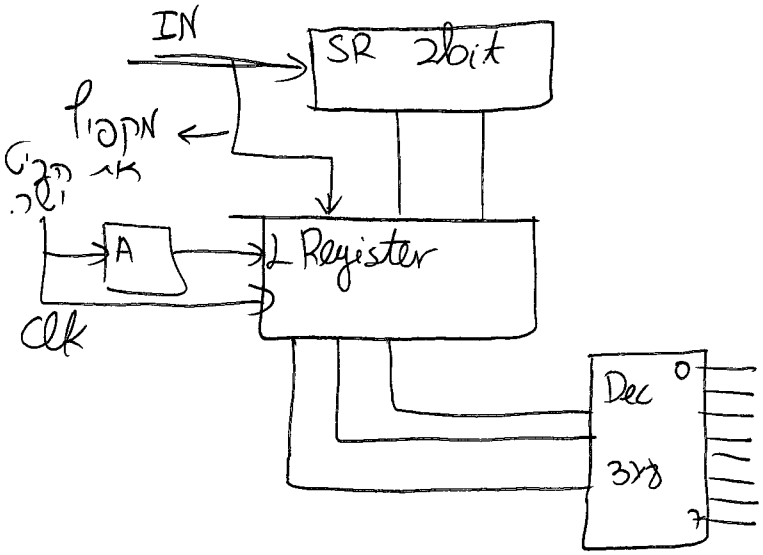


מקבלים סדרת ביטים

עבור למשל: In 001010011010

Out 0011222333222...

סלולה המשרים מקבל "שלשות" המגדירה כל פעם מאיזה
מכונה כל ביט
כל פעם-8



כולל אוכלוסיה שמעידה עמדה שבו ForwardA=01
ForwardB=10
Flush=1

```
add $2, $6, $7
add $3 $4 $5
add $1 $2 $3
beq $9 $9
```