אוניברסיטת בר-אילן (ע"ר)
הפקולטה להנדסה
רמת-גן 52900

BAR-ILAN UNIVERSITY (RA)
Faculty of Engineering
Ramat-Gan 52900, Israel

Tel: 03-5317722

engbi@mail.biu.ac.il

תכן לוגי

תשע"ח סמסטר קיץ מועד א'

83-253

מרצה: פרופ' שמואל וימר

מתרגלים: מר בנימין פרנקל ומר גילעד פלן

- **יש לקרוא היטב את ההוראות.**
- **חובה לענות על כל השאלות.**
- **סך הנקודות בבחינה הוא 110 אך הציון הסופי בבחינה לא יעלה על 100.**
- **יש לנמק את כל תשובותיכם.**
- יש לשרטט באופן ברור, להוסיף הסבר מילולי, ובמידת הצורך לצייר את החלק הרלוונטי במחברת הבחינה.
- יש להקפיד על כתב יד קריא!
- **חומר עזר מותר בשימוש**: מחשבון, ספר הקורס ושקפים מההרצאות בלבד (כולל הערות על גביהם).
- **משך הבחינה:** שלוש שעות.
- **יש לצרף את שאלוני הבחינה למחברת!**

בהצלחה!

<u>שאלה מספר 1</u> <u>(50 נקודות):</u>

בשאלה זו נרצה לשנות את תכן ה- single cycle MIPS כך שיתמוך בפקודות חדשות:

עליך לשנות את מבנה ה- single cycle MIPS המתואר בתרשים מספר 1 כך שיתמוך בשתי הפקודות LB (Load Byte) ו- LBU (Load Byte Unsigned), כאשר הפקודות מקודדות באופן הבא:

```
lb rt, offset(base)
```

LB 1 0 0 0 0 0	base	rt	offset

```
lbu rt, offset(base)
```

LBU 1 0 0 1 0 0	base	rt	offset

כל אחת מהפקודות לוקחת את ערך ה- immediate המופיע בשדה ה- offset, מבצעת לו-sign extension ל- 32 סיביות וסוכמת אותו ביחד עם הערך האגור ברגיסטר 'base'. הסכום המתקבל משמש ככתובת של Byte בודד שנמצא בזיכרון, כאשר את המידע שנמצא ב- Byte הזה הפקודה תשמור ברגיסטר rt ב- register-file.

ההבדל בין שתי הפקודות הוא ביחס למידע שנמצא באותו - Byte שהפקודה מביאה מהזיכרון. הפקודה LB תתייחס למידע כ- signed value, ולכן תעשה לו sign-extension לאורך של 32 סיביות לפי ה- MSB של אותו ה- Byte, ואילו הפקודה LBU תתייחס למידע כ- unsigned value, ולכן פשוט תרפד אותו משמאל ב- 24 אפסים.

א. עליך להוסיף על גבי **תרשים מספר 1** רכיבים ב- datapath הדרושים לצורך תמיכה בשתי הפקודות הנ"ל. תן שם לכל אות בקרה (control signal) חדש שנדרש להוסיף, והראה כיצד יש להוסיף או לשנות את ה- Decoder הראשי על מנת לתמוך בשתי הפקודות החדשות.

- כעת, נרצה לשנות את תכן ה- single cycle MIPS כך שיתמוך בפקודה חדשה אחרת.

הפקודה BGEZAL (Branch on Greater or Equal to Zero and Link) הינה אחת מתוך Instruction Class (סוג פקודות) שנקרא REGIMM.

הפקודה BAGEZAL מקודדת באופן הבא:

```
bgezal rs, offset
```

REGIMM 0 0 0 0 0 1	rs	BGEZAL 1 0 0 0 0	offset

שים-לב, כי כאשר שדה ה- opcode מכיל את הערך REGIMM אזי המעבד יודע שהוא צריך להסתכל על שדה ה- rt בשביל לפענח איזו פקודה בדיוק הוא צריך לבצע. למעשה, בפקודות מסוג REGIMM שדה ה- rt מהווה הרחבה של ה- opcode.

הפקודה קוראת את התוכן האגור ברגיסטר rs ואם הוא גדול או שווה לאפס אז תתבצע קפיצה לפקודה הרחוקה במרחק של 'offset' מילים מהמיקום הנוכחי של ה- PC+4 (בדומה למה שעושה פקודת bne). אחרת, המעבד ימשיך לבצע את הפקודה הבאה בתכנית.

בנוסף, ללא קשר לערך האגור ב- rs, הפקודה כותבת את הערך של PC+4 לתוך רגיסטר 31 ($ra) ובכך מספקת את תכונת ה- "and Link".

ב. בדומה לסעיף הקודם, עליך להוסיף על גבי **תרשים מספר 2** רכיבים ב- datapath הדרושים לצורך תמיכה בפקודה הנ"ל. תן שם לכל אות בקרה (control signal) חדש שנדרש להוסיף, והראה כיצד יש להוסיף או לשנות את ה- Decoder הראשי על מנת לתמוך בפקודה החדשה.

ג. במקום לממש את הפקודה bgezal באופן ישיר בחומרה, הועלה רעיון לממש אותה בתכנה (software) כ- pseudoinstruction. רשום רצף פקודות מתוך כל הפקודות שנלמדו בקורס על מנת לממש את הפקודה bgezal.

פתרון:

```
    slt   $t0   $rs   $zero
    bne   $t0   $zero L1
    jal   offset
L1:
```

או לחלופין:

```
    slt   $t0   $zero $rs
    beq   $t0   $zero L1
    jal   offset
L1:
```

ד. נניח שמימוש הפקודה bgezal באופן תכנתי (software implementation) לוקח 3 מחזורי שעון, בעוד שמימוש הפקודה באופן חומרתי (hardware implementation) לוקח מחזור שעון אחד, אבל מחייב להגדיל את מחזור השעון של המעבד ב- 10%. כמה פקודות bgezal צריכה תכנית לבצע ביחס לשאר הפקודות בתכנית (באחוזים) כדי שהמימוש החומרתי יהיה כדאי מבחינת זמן ביצוע (execution time)?

זמן הביצוע של התכנית הוא:

$$Execution\ time = IC \times CPI \times CCT$$

נסמן את המרכיבים של זמן הביצוע של המימוש ב- software כך: $IC_s \times CPI_s \times CCT_s$

ובאופן דומה, נסמן את המרכיבים של זמן הביצוע של המימוש ב- hardware: $IC_h \times CPI_h \times CCT_h$.

נסמן ב- I את מספר הפקודות שאינן bgezal, ונסמן ב- B את מספר הפקודות שהן bgezal.

נבדוק מתי זמן הביצוע משתווה:

$$\mathrm{IC}_s \times CPI_s \times CCT_s = \mathrm{IC}_h \times CPI_h \times CCT_h$$

$$(I + 3B) \times CCT_s = (I + B) \times 1.1 \cdot CCT_s$$

$$(I + 3B) = (I + B) \times 1.1$$

$$1.9B = 0.1I$$

$$\frac{B}{I} = \frac{0.1}{1.9} = \frac{1}{19} \cong 5.26\%$$

ולכן, כדי שיהיה כדאי להשתמש בפתרון ה- hardware צריך להתקיים שהיחס בין פקודות ה-bgezal שמתבצעות במהלך התכנית לבין שאר הפקודות יהיה גדול מ- 5.26%

<u>שאלה מספר 2</u> <u>(30 נקודות):</u>

בשאלה זו נעסוק בטיפול ב- Hazards הקיימים בארכיטקטורת Pipelined MIPS.

נתון רצף פקודות האסמבלי הבא:

```
add   $3   $2   $1
sub   $10  $3   $1
and   $12  $3   $10
or    $4   $3   $10
sw    $7   0($4)
sw    $4   0($3)
```

כאשר שלב ה- Instruction Fetch של הפקודה הראשונה (add) מתבצע במחזור שעון מספר 1.

א. כיצד יש לתזמן את הרכיבים השונים במעבד ה- Pipelined MIPS על מנת שלא תיווצר בעיה בביצוע הפקודות add ו- or כפי שהן מופיעות ברצף הנ"ל? הסבר היטב!

יש לתזמן את ה- Register File כך שידגום בירידת השעון (negative-edge triggered), ואת כל שאר הרכיבים המתוזמנים יש לתזמן כך שידגמו בעליית השעון (positive-edge triggered). באופן זה, פעולת ה- WB של פקודת ה- add תתבצע בחצי הראשון של מחזור השעון החמישי (משום שהדגימה אל תוך ה- Register File תתבצע בירידת השעון), והקריאה של הערך העדכני תתבצע ע"י פקודת ה- or במחצית השנייה של מחזור השעון החמישי (בשלב ה- ID, כאשר הדגימה אל ה- pipeline-register תתבצע בעליית השעון של סוף מחזור השעון החמישי – תחילת מחזור השעון השישי). אם כל הרכיבים היו מתוזמנים כך שידגמו בעליית שעון (positive-edge triggered), הרי שהפקודה or לא הייתה קוראת את הערך המעודכן של $3, משום שהוא היה מתעדכן רק **בסוף** זמן המחזור החמישי, עם עליית השעון.

ב. אילו Hazards ישנם ברצף הפקודות הנ"ל? פרט והסבר!

1. פקודת ה- sub מנסה לקרוא ערך מ- $3, כאשר פקודת ה- add עדיין לא עשתה לו WB. זהו hazard מסוג 1a על פי שקף מספר 6 בהרצאה.
2. פקודת and מנסה לקרוא ערך מ- $10 כאשר פקודת ה- sub עדיין לא עשתה לו WB. זהו hazard מסוג 1b.
3. פקודת ה- and מנסה לקרוא ערך מ- $3, כאשר פקודת ה- add עדיין לא עשתה לו WB. זהו hazard מסוג 2a.
4. פקודת or מנסה לקרוא ערך מ- $10 כאשר פקודת ה- sub עדיין לא עשתה לו WB. זהו hazard מסוג 2b.
5. פקודת or מנסה לקרוא ערך מ- $3 כאשר פקודת add עושה WB לאותו רגיסטר באותו זמן מחזור. הבעיה יכולה להיפתר ע"י תזמון השעון כפי שהוסבר בסעיף א'.

6. פקודת ה- sw הראשונה מנסה לקרוא ערך מ- $4 כאשר פקודת ה- or עדיין לא עשתה לו WB. זהו hazard מסוג 1a.
7. פקודת ה- sw השנייה מנסה לקרוא ערך מ- $4 כאשר פקודת ה- or עדיין לא עשתה לו WB. זהו hazard מסוג 2b.

העזר בתרשים מספר 3 לצורך מענה על הסעיפים הבאים:

ג. ברצוננו לפתור את כל ה- Hazards שמצאת בסעיף הקודם בעזרת ה- Forwarding Unit. בטבלה הבאה עליך לרשום עבור כל מחזור שעון את הערכים של קווי הבקרה המנתבים את זרימת המידע בשלב ה- Execution. אם ישנם Don't-cares ציין זאת בטבלה!

Clock Cycle	ForwardA	ForwardB	ALUsrc	RegDst
3	0	0	0	1
4	2	0	0	1
5	1	2	0	1
6	0	1	0	1
7	2	0	1	X
8	0	1	1	X

- מהנדס חד-עין שם לב כי המבנה הנוכחי של יחידת ה- Forwarding Unit אינו יכול לטפל במקרה שבו מופיע רצף הפקודות הבא:

```
lw    $2    0($4)
sw    $2    0($6)
```

ד. עליך להוסיף על גבי **תרשים מספר 3** רכיבים ב- datapath הדרושים לצורך טיפול ב- Hazard הנ"ל בעזרת ה- Forwarding Unit. תן שם לכל אות בקרה (control signal) חדש שנדרש להוסיף.

<u>שאלה מספר 3</u> (30 נקודות):

בשאלה זו נעסוק בזיכרון מטמון (cache).

בסעיפים הבאים נעסוק ב- direct mapped cache בעל 64 בלוקים, כאשר גודל כל בלוק הוא 16B.

הנח כי רזולוציית המיעון היא 1 Byte. הזיכרון הראשי מכיל 2^{24} B.

א. שרטט את מבנה הכתובת וחלוקתה לשדות השונים (tag/index/offset). ציין מספרי סיביות וכן את גודלי השדות.

tag			index			offset		
23		10	9		4	3		0

ב. כמה בלוקים מכיל הזיכרון הראשי?

$$2^{24}/_{2^4} = 2^{20} Blocks$$

ג. לאיזה set בזיכרון המטמון ממופת הכתובת 0xDECADE מהזיכרון הראשי?

0xDECADE = 1101 1110 1100 1010 1101 1110_2
Set number = bits 4–9 = $101101 = 45_{10}$

ד. כעת, הנח כי זיכרון המטמון ריק. עבור רצף הגישות הבא, רשום איזה אירוע יתרחש בכל אחת מהגישות לזיכרון. בתשובתך, התייחס לאירוע hit/miss, והסבר איזו תכונת לוקאליות גרמה לכל hit, וכן מהי הסיבה שגרמה לכל- miss (האם הבלוק בזיכרון המטמון היה ריק, או שהוא היה מלא בערכים שאינם מתאימים לגישה הנוכחית לזיכרון).

Address	**Event**
0xDECADE	Compulsory Miss
0xDECAD8	Spatial locality hit (same block as above)
0xDECAE8	Compulsory Miss (different block)
0xBECADE	Conflict Miss (same block; different tag)
0xDECADE	Conflict Miss (same block; different tag)
0xDECAD8	Spatial locality hit (same block as above)
0xDECADE	Temporal locality hit (same byte as above)

ה. בסעיפים הבאים, נעסוק במחשב בעל זיכרון מטמון fully associative cache בעל 32 בלוקים, כאשר כל בלוק הוא 64B. הזיכרון הראשי מכיל 2^{16} B. כמה בלוקים יש בזיכרון הראשי?

$$2^{16}/_{2^6} = 2^{10} Blocks$$

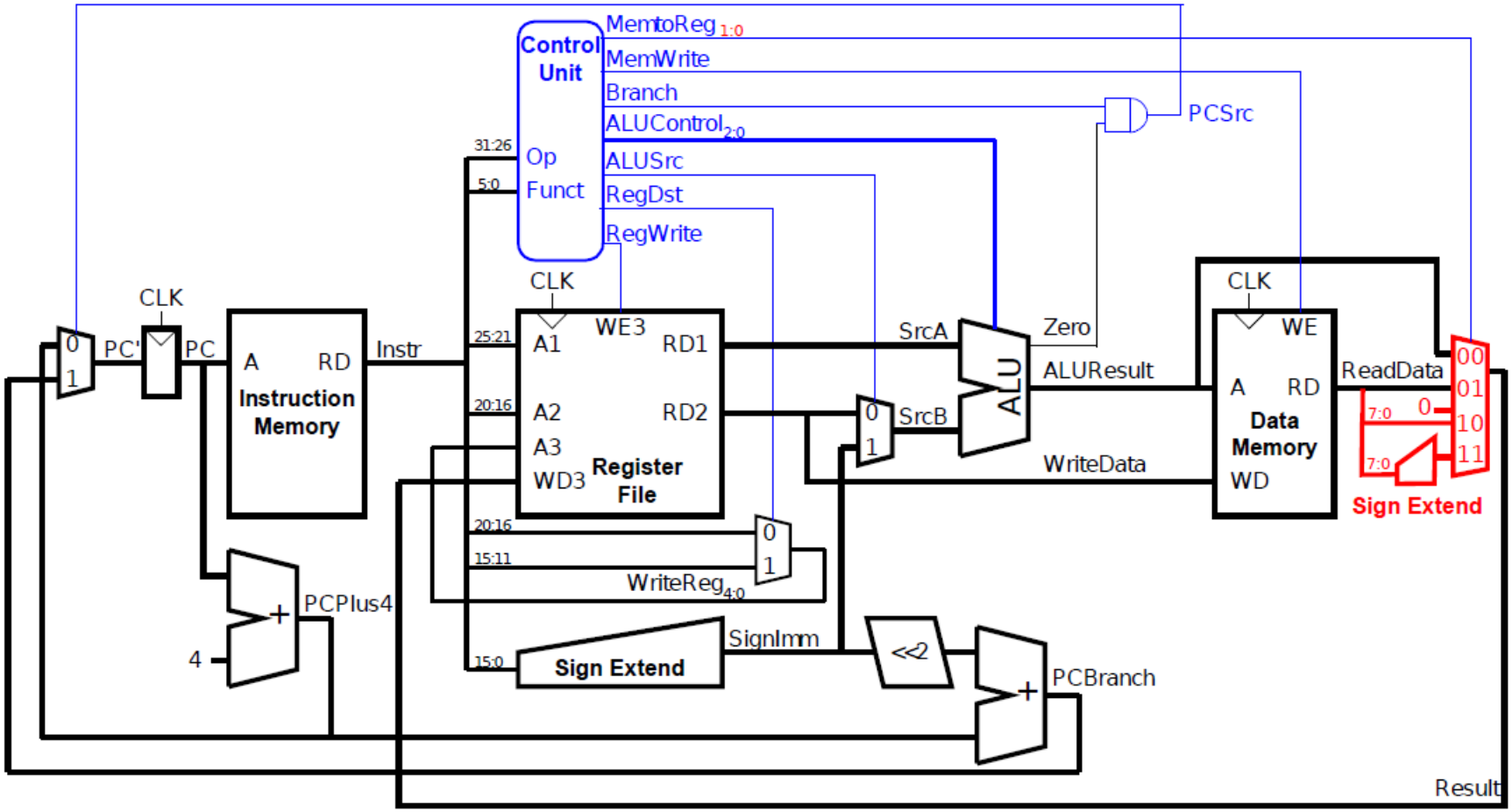
ו. שרטט את מבנה הכתובת (עבור המחשב מהסעיף הקודם) וחלוקתה לשדות השונים (tag/index/offset). ציין מספרי סיביות וכן את גודלי השדות.

tag			offset		
15		6	5		0

ז. לאיזה set בזיכרון המטמון ממופת הכתובת 0xF00D מהזיכרון הראשי?

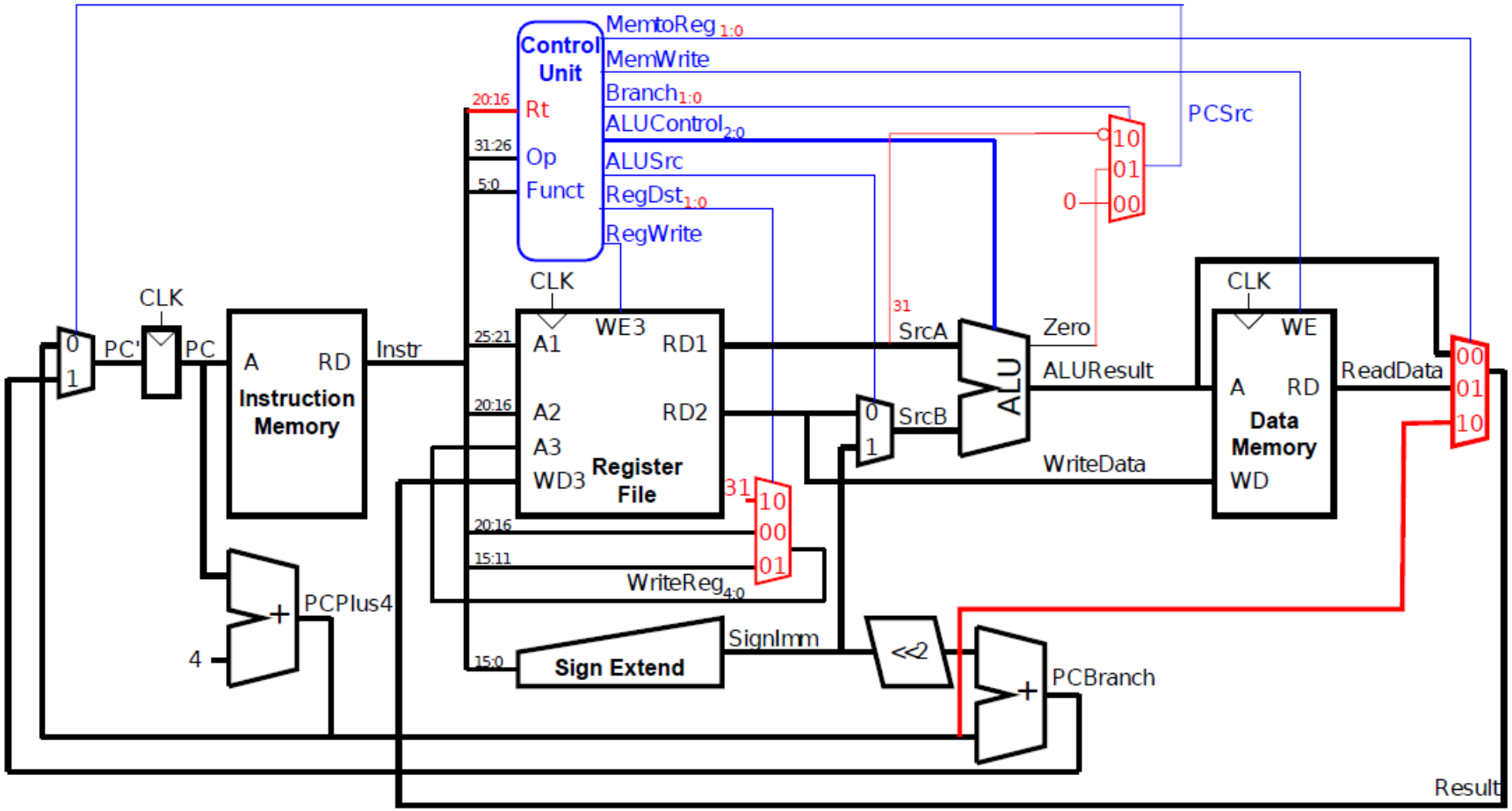
ל- Fully associative cache יש אך ורק set אחד ויחיד, שהכול מתמפה אליו.

תרשים מספר 1



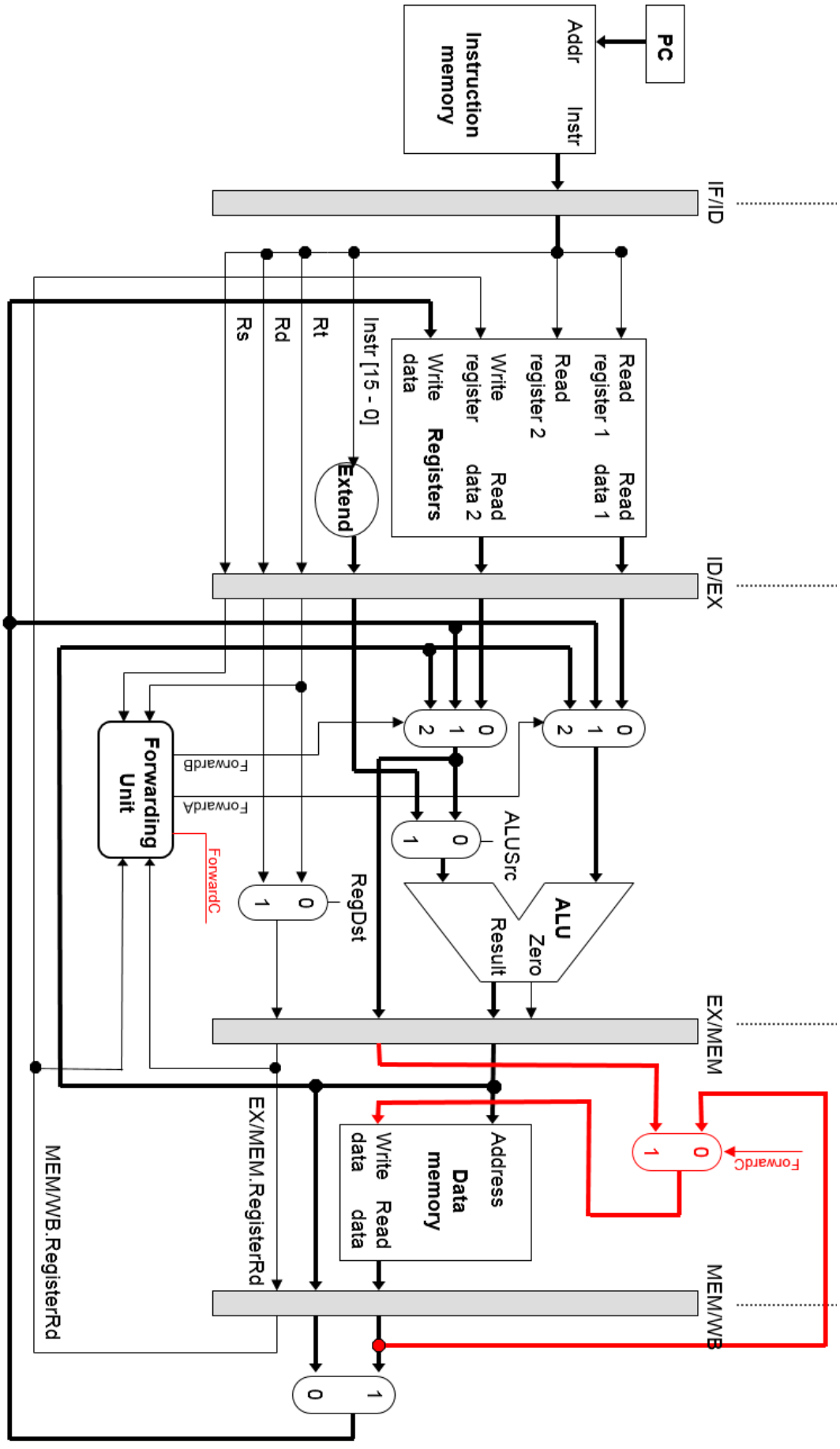
Inst.	OP	RegWrite	RegDst	ALUSrc	Branch	MemWrite	MemToReg	ALUOp
R-type	000000	1	1	0	0	0	00	1-
lw	100011	1	0	1	0	0	01	00
sw	101011	0	-	1	0	1	--	00
beq	000100	0	-	0	1	0	--	01
lb	100000	1	0	1	0	0	11	00
lbu	100100	1	0	1	0	0	10	00

תרשים מספר 2



Inst.	OP	Rt	RegWrite	RegDst	ALUSrc	Branch	MemWrite	MemToReg	ALUOp
R-type	000000	-	1	01	0	00	0	00	1-
lw	100011	-	1	00	1	00	0	01	00
sw	101011	-	0	--	1	00	1	--	00
beq	000100	-	0	--	0	01	0	--	01
bgezal	000001	10000	1	10	-	10	00	10	--

תרשים מספר 3


PC
Addr
Instr
Instruction memory
IF/ID
Rs
Rd
Rt
Instr [15 - 0]
Extend
Read register 1
Read register 2
Write register
Write data
Read data 1
Read data 2
Registers
ID/EX
Forward B
Forward A
Forwarding Unit
ForwardC
RegDst
ALUSrc
ALU
Zero
Result
EX/MEM
Forward C
Address
Data memory
Write data
Read data
MEM/WB
EX/MEM.RegisterRd
MEM/WB.RegisterRd