מבנה מחשבים-הרצאה מספר 7

<u>הגדרה:</u> שפת מכונה הינה שפה המורכבת מביטים,כך שכל רצף של ביטים מייצג פעולה בסיסית למחשב. לדוגמא בשפת המכונה של mips הרצף 00011 00001 00010 000000 100000 00000 יחבר לנו בין רגיסטרים מספר 1 ומספר 2 והתשובה הוא ישים ברגיסטר ישים 3(יש למחשב שלנו כמה רגיסטרים.נגיע לזה בהמשך...)

<u>הגדרה:</u> שפת סף(יש הקוראים לזה שפת אסמבלי) הינה שפה תיכנות שמבצעת פעולות באופן לשפת המכונה חיבור בין 2 רגיסטרים $9 ו$8 וישים ברגיסטר $4 בmips יבצע כך add $4 $8 $9.

בניגוד לשפות עילית(כגון JAVA וC) שפת סף יש בה פעולות הרבה יותר פשוטות,יותר קשה לתכנת איתו והמעבר לשפת המכונה הוא חח"ע.כמוכן,שפות העילית מורכבות משפות סף.

בהרצאה מספר 8 ובהרצאה מספר 9 אנו נלמד איך שפת המכונה פועל בצורה מוחשית(מרצף של הביטים לביצוע הפעולות עצמם).

מה ההבדל בין אסמבלר לאסמבלי? האסמבלר הוא ה**תוכנית** שממיר מהאסמבלי לשפת המכונה והאסמבלי הוא השם של ה**שפה**.

בכן,אנו נלמד בהרצאה זאת על שפת הסף של הmips אשר היא בפרט חלק מטכלוגיית CISC(כפי שהוסבר בשיעור הראשון).

בmips יש לנו 32 רגיסטרים , מסוג של D Type ולכל רגיסטר יש 32 דלגלגים.הרגיסטרים קשורים לשפת העילית(נניח בשפת C) שבה היא הmips שומש כשפת הסף שלה.כמוכן לכל מתודה שאנו עושים בשפת C אנו מייצרים את הרגיסטרים המיוחדים "שלה" כאשר אנו מפעילים את המתודה..ראה טבלה שלפניכם על סוגי הרגיסטרים(של הMips):

Name	Register number	Usage
$zero	0	הקבוע 0
$v0-$v1	2-3	ערכי ההחזרה במתודה
$a0-$a3	4-7	הארגומנטים של המתודה
$t0-$t7	8-15	משתני זבל (שלא הצהרנו עליהם בשפת העילית)
$s0-$s7	16-23	משתנים שהצהרנו עליהם בשפת העילית
$t8-$t9	24-25	אותם רגיסטרים כמו של 8-15
$gp	28	משתנה שמכיל כתובת(מצביע) למשתנה גלובלי(משתנה שהוא מחוץ למתודה)
$sp	29	משתנה שיצביע על המחסנית של הstack(ראה למטה)
$fp	30	ראו בתרגול
$ra	31	הכתובת בזיכרון אליו הוא "הולך" ברגע שהוא יסיים את המתודה בו הוא פועל

רגיסטרים מספר 1 שנקרה בשם $at שקשור לאסמבלר(ולא למעבד) ו27-28 לא צריך לדעת(קשורים למערכת ההפעלה).

לבינתיים,נשתמש רק ברגיסטרים $t0-$t7.נחזור יותר מאוחר על כל שאר הרגיסטרים.

<u>הmemory</u>

הזיכרון שלנו(RAM) ניתן להגיד שהוא בעצם מערך אחד ענקי בגודל של 2^{32} תאים כך שבכל תא במערך,ישנו 8 ביטים=1 בית.

נאמר כי 4 בית=מילה=32 ביטים.נשים לב כי בעצם מילה אחת מייצג מספר שלם מסוג int (סה"ך יש 2^{32} מספרים מסוג int) וכי תו אחד(בטבלת ASCII) הוא מיוצג ע"י בית אחד(ולכן יש סה"ך 2^8 תווים בASCII).

הערות:
1)נאמר כי אנו רוצים לגשת למקום הk בזיכרון שלנו כאשר רוצים להגיע לתא הk-ית של המערך של הזכרון(המקום הראשון הוא התא מספר 0)
2)נוכל גם להתבונן על הזכרון שלנו כאל מערך של מילים בצורה הבאה: המילה הk-ית במערך היה שירשור של הבתים(לפי הסדר שמשמאל לימין) שבמקומות 4*k,4*k+1,4*k+2,4*k+3 במערך שלנו (המיוצג לפי ביטים).
לדוגמא: אם במקום ה12 במערך של ביטים יש לנו הבית 10101010 במקום ה13 יש את 01010101 במקום ה14 יש את יש את 00000000 ובמקום ה15 יש את 11111111 אזי המקום ה4 במערך של המילים ימצא המילה:
10101010010101010000000011111111
3)בשפת C ניתן להגדיר מערך לפי הגודל שלו ולפי משתנה x שיכיל בתוכו את הכתובת(הוא מצביע) של האיבר הראשון שבמערך.x+1 היה הבית השני במערך(x+4 היה המילה השניה במערך-כמוכן כל מערך בC הוא בעצם מערך של ביתים).
4)יש לא להתבלבל בין משתנה שהוא מצביע למערך למשתנה שמכיל ערך מספרי.אם x הוא משתנה שמכיל ערך מספרי אזי x+1 היה המספר העוקב למשתנה ואם x הוא מצביע לאיבר מספר i מערך של בתים אזי x+1 יצביע על המקום הi+1 במערך

הפעולות בmips

pc counter:
בmips כל פעולה מיוצגת ע"י מילה בודדת,ואנו בעצם מייבאים את הפקודות האלו מהתוכנית למעבד שלנו.התוכנית שלנו היא בעצם חלק מהזיכרון שלנו,והוא מיוצג ע"י מערך של ביטים .הPC שלנו הוא בעצם מצביע לכתובת של הפקודה שמתבצעת.לכן כדי לגשת לפעולה הבאה בתוכנית שלנו אנו חייבים לקדם ב4 את הPC שלנו.
ישנם כמה סוגי פעולות:

פעולות R Format:
פעולות R-Format אלו פעולות אריטמטיות בין שלושה רגיסטרים,כל אחד מהם הוא מהסוג הבא:

opcode	rs	rt	rd	shamt	funct

כאשר:
opcode- רצף של 6 ביטים המתארים סוגים שונים של פעולות.בכל פעולות הR-Type מתקיים כי opcode=000000
rs,rt,rd-אלו הם כתובות לרגיסטרים של הmips(ראה למעלה) ולכן כל אחד מהם הוא רצף של 5 ביטים.אם נניח שהפעולה שלנו הוא פעולת fun ונניח שrs,rt,rd מצביעים על רגיסטרים x,y,z(בהתאמה) אזי אנו בעצם עושים את הפעולה x=y fun z .
shamt-רצף של 5 ביטים אשר עושים ,בנוסף לכל הפעולות shift right או shift left לרגיסטרים שלנו(לא צריך לדעת את זה למבחן).
funct- רצף של 6 ביטים,שמחליט איזה פעולה לבצע מתוך כל פעולות הR Type הקיימות

דוגמאות חשובות לR-Type:
1) **add $t0 $t1 $t2 -** עושה את הפעולה $t0=$t1+$t2 (כלומר החיבור של כל ביט-ביט מהרגיסטרים,$t1 ו$t2 נציב ברגיסטר $t3) הוא מסוג R Type.נניח שאנו רוצים לחבר את רגסיטר 10 עם רגיסטר 11 ולשים את התוצאה ברגיסטר 13.אזי הפעולה שתבוצעה תהיה:

13	11	10			
000000	01101	01011	01010	00000	100000

<u>יש לזכור כי $t0,$t1 אלו רגיסטרים שיש לנו בmips ולא הערכים שיש בכתובות עצמם!!!(בדוגמא של למעלה אנחנו לא מחברים בין 10 ו11 ושמים את הערך ברגיסטר 13)</u>

בצורה דומה, גם sub $t0 $t1 $t2 וגם mult $t0 $t1 $t2 הוא מסוג R-Type.

2)**<u>slt $t0 $t1 $t2</u>**- אם $t1<$t2(כלומר אם הערך שיש בתוך רגיסטר $t1 יותר גדולה מהערך שיש ברגיסטר $t2) עשה ש$t0=1 אחרת $t1=1(לא נראה את היצוג הפנימי של הרגיסטר)

3)**<u>move $t0 $t1-</u>** עשה ש$t1=$t0(שקול ל add $t0 $t1 $zero)

4)**<u>and $t0 $t1 $t2-</u>** יעשה פעולת and בין שני רגיסטרים $t1,$t2(כלומר פעולת and בין כל שני רגיסטרים) וישים ב$t3 (בצורה דומה אפשר להגדיר פעולת or,xor וג')

5)**<u>jr $t0-</u>** "יקפוץ" לפעולה שבערך של $t0(כאשר הערך שב$t0 מייצג פעולה מתוך התוכנית שלנו)

<u>פעולות I Format:</u>

פעולות I-Format אלו פעולות בין שני רגיסטרים וקבוע,והם נראים בצורה המיוחדת הבאה:

opcode	rs	rt	offset/ immediate

כאשר opcode עם 6 ביטים יכול להיות הכל,rs וrt הם רגיסטרים עם 5 ביטים והקבוע(**immediate**) הוא עם 16 ביטים.

דוגמאות חשובות לפעולות I-Format:

1) **<u>muli $t0 $t1 x-</u>**מכפיל את המספר שברגיסטר $t1 בx ומציבים ל$t0

2) **<u>addi $29, $29, 4-</u>**מחבר את הערך שברגיסטר $29 עם הקבוע 4

3) **<u>lw $t0, x($t1)</u>** (ולא כותבים x $t1 $t0 lw)–נסמן את המערך של הזכרון בmem .הפעולה בעצם תוציא לנו את המילה שמורכב מהבתים
mem[x+$t1],mem[x+$t1+1],mem[x+$t1+2],mem[x+$t1+3] (x+$t1 הוא בעצם רגיסטר,ולכן הוא מייצג מספר כלשהו,נניח y,ומתקיים בעצם mem[y]==mem[x+$t1]) וישים את המילה הנ"ל לרגיסטר $t0(כלומר הפעולה טוענת מילה מהזיכרון ושמה לתוך רגיסטר)

4)**<u>beq $t1 $t2 x-</u>** אם $t0==$t1 לך x+1 פעולות קדימה(נניח שכבר כתבנו את הקוד בשפת מכונה,כך שהפעולה beq $t1 $t2 x נמצאת בשורת קוד מספר 60.אם $t0=!$t1 לך לפקודה הבאה(כמו שבדרך כלל עושים) . אם $t0==$t1 קפוץ לשורת הקוד x+1+60)
ברור שהפעולה beq $t1 $t2 0 בעצם לא תשפיע לנו על שום דבר,בין אם הרגיסטרים שויים ולבין אם לא תמיד ניקפוץ לפעולה הבאה.

<u>הערה:</u> ניתן לעשות גם beq $t1 $t2 fun כלומר אם $t0==$t1 קפוץ לשורה בו מופיע fun (ראה דוגמא)

5)**<u>sw $t0 x($t1)</u>**-יישמור בתוך
mem[x+$t1],mem[x+$t1+1],mem[x+$t1+2],mem[x+$t1+3] אתהערך שיש ברגיסטר $t0([x+$t1]mem היה את את 8 הביטים הראשונים של הרגיסטר [x+$t1+1]mem היה את את 8 הביטים הבאים של הרגיסטר וכ')

6)**<u>lb $t0,x($t1)</u>**-טוען לנו את הבית [x+$t1]mem לתוך 8 הביטים הכי ימניים של $t1

7)**sb $t0,x($t1) -** טוען לנו מהבית הכי ימני של $t1 לתוך mem[x+$t1]
8)**bne $t0,$t1,x -** אם $t0=!$t1 לך x+1 פעולות קדימה אחרת קפוץ לפקודה הבאה.
הערה: ניתן לעשות גם bne $t1 $t2 fun כלומר אם $t1=!$t0 קפוץ לשורה בו מופיע fun (ראה דוגמא)
9)פעולות slti ori andi וג'–אותה קונוטוציה כמו עם add ועם addi
10)**lui $t0 x**- ישים ברגיסטר $t0 את המספר 2^{16} * x. כלומר אם נתון לנו המספר x ביצוג הביניירי שלנו אזי אנו נציב ב$t0 את x כאשר הזזנו אותו 16 ביטים שמאלה(16 הביטים שה"כנסנו" ל-x הם אפסים).השימוש העיקרי של פעולה זאת היא ליצור קבועים שגדולים מ 2^{16},שכן בפעולות הI-Type כל הקבועים הם קטנים מ 2^{16} (כי יש רק 16 ביטים)
11) **blt $t0 $t1 x-** אם $t0<$t1 לך x+1 פעולות קדימה אחרת קפוץ לפקודה הבאה.

פעולות jump:
פעולות jump אלו פעולות שיש לוהם רק קבוע ,והם נראים בצורה המיוחדת הבאה:

opcode	trarget

כאשר opcode הוא 6 ביטים וtarget הוא עם 26 ביטים
דוגמאות חשובות:
1)**j x-** יקפוץ לפקודה שנמצא במקום הx(בניגוד לbranch שבו אנו קופצים "קדימה" תמיד)
הערה: ניתן לעשות גם j fun ,כלומר קפוץ לשורה בו מופיע fun
2)**jal x –** יקפוץ לפקודה שנמצא במקום הx ו-שומר ברגיסטר $ra את הכתובת של הפקודה שאחריו.
הערה: ניתן לעשות גם jal fun ,כלומר קפוץ לשורה בו מופיע fun ושמור ברגיסטר $ra את הכתובת שאחרי הפקודה jal x

הערות נוספות:
1)פעולת הjal נועדה בעיקרון לפתור לנו את הבעייה שבו יש לנו מתודה אחת(נניח f) שמשתמשת במתודה אחרת(נניח g),שכן בסיום השימוש של המתודה של g אנו חייבים לחזור לפונקציה המקורית שלנו f .משום ש$ra מצביע עתה על שורה שאחרי פקודת הjal אזי כאשר נעשה jr $ra אנו נחזור לפונקצייה המקורית שלנו f לשורה בה היא אמורה להיות.
2) לכן נרצה שכל פעולה תסתיים בסוף בפקודה jr $ra