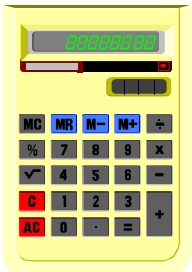
מאיה שמעון



קצת על פעולות אריתמטיות

ADD op1, op2

חיבור op1 ו- op2 והצבת התוצאה לתוך op1 (op1 = op1 + op2).
יש לשמור על כלל התאמת האופרנדים, כלומר ש- op1 ו- op2 יהיו אופרנדים בגודל זהה.
דוגמאות:

```
ADD BL, CH
ADD Var, 100  (Var – משתנה זכרון)
ADD AX, 800
```

אופרנדים מותרים:

```
ADD r, r or m
ADD m,r
ADD r or m,imm
```

הערה:
הפקודה טובה לשימוש במספרים מסומנים ולא מסומנים.

SUB op1, op2

חיסור op2 מ- op1 והצבת התוצאה לתוך op1. (op1 = op1 - op2)
יש לשמור על כלל התאמת האופרנדים, כלומר ש- op1 ו- op2 יהיו אופרנדים בגודל זהה.
דוגמאות:

```
SUB BL, CH
SUB Var, 100  (Var – משתנה זכרון)
SUB AX, 800
```

אופרנדים מותרים:

```
SUB r, r or m
SUB m,r
SUB r or m,imm
```

הערה:
הפקודה טובה לשימוש במספרים מסומנים ולא מסומנים.

INC op

הגדלת op ב - 1.
דוגמאות:

```
INC AX
INC Var
```

אופרנדים מותרים:

```
INC m or r
```

הערות:
1. הפקודה נכונה לשימוש במספרים מסומנים ולא מסומנים.
2. פקודה זו אינה משנה את דגל ה- carry.

DEC op

הקטנת op ב - 1.
דוגמאות:

```
DEC AX
DEC Var
```

אופרנדים מותרים:

```
DEC m or r
```

הערות:
1. הפקודה נכונה לשימוש במספרים מסומנים ולא מסומנים.
2. פקודה זו אינה משנה את דגל ה- carry.

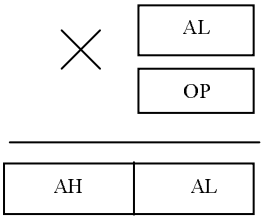
CMP op1, op2

הפקודה מבצעת חיסור op2 מ- op1 אך לא מציבה את התוצאה לתוך op1.
אם כן מה הטעם? הפקודה משפיעה על אוגר הדגלים (אשר מושפע כהרגלו מהפעולה האריתמטית/לוגית האחרונה שהתבצעה) ולפי השינוי בדגלים ניתן לבדוק איזה אופרנד גדול יותר או אם הם בכלל שווים (אם הם שווים קל להבין שה- ZF ידלוק לאחר פעולת CMP).
יש לשמור על כלל התאמת האופרנדים, כלומר ש- op1 ו- op2 יהיו אופרנדים בגודל זהה.

MUL op

<u>אם op הוא אופרנד בן 8 ביטים</u>

AL מוכפל בתוכן של op והתוצאה נזרקת ל- AX.

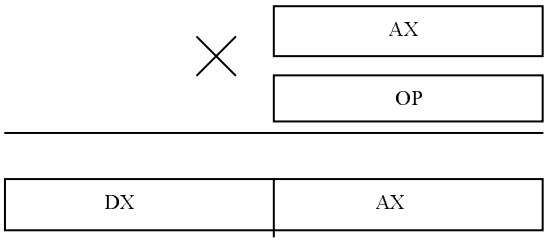


דוגמאות:

MUL CL
MUL Var
כאשר Var הוא משתנה זכרון בן בית אחד

<u>אם op הוא אופרנד בן 16 ביטים</u>

AX מוכפל בתוכן של op והתוצאה נזרקת לאוגרים AX, DX באופן הבא: 16 הסיביות המשמעותיות יותר של התוצאה יכנסו ל- DX ו- 16 הסיביות הפחות משמעויות של התוצאה יכנסו ל- AX.



דוגמאות:

MUL BX
MUL Var
כאשר Var הוא משתנה זכרון בן שני בתים

<u>אופרנדים מותרים:</u>

MUL r or m

הערות:

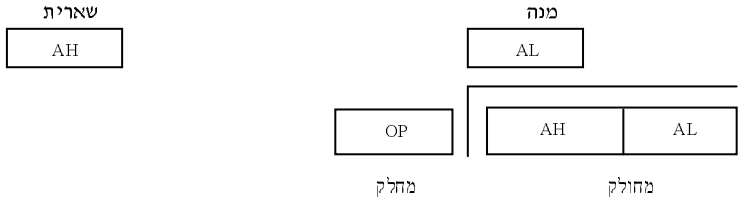
1. פקודה זו טובה לשימוש במספרים לא מסומנים בלבד. בכדי לבצע פעולת כפל עם מספרים מסומנים יש להשתמש בפקודה IMUL.
2. במעבד 386 יש אפשרות לבצע פעולת כפל כאשר op הוא אופרנד בן 32 ביט. אז op יוכפל ב- EAX ותוצאת הכפל תיזרק ל: EDX:EAX.

DIV op

אם op הוא אופרנד בן 8 ביטים

חילוק AX ב- op. המנה נזרקת ל- AL והשארית ל- AH.

קצת עצוב:
נסו לראות מה קורה אם המנה לא ניתנת לייצוג על ידי 8 סיביות (ומתי זה יקרה). האם יש לכם רעיון לפתור בעיה זו?



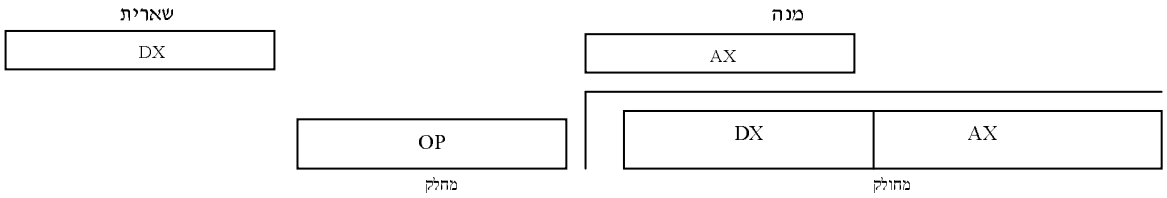
דוגמאות:

DIV CL
DIV Var
כאשר Var הוא משתנה זיכרון בן בית אחד

אם op הוא אופרנד בן 16 ביטים

חילוק DX:AX ב- op. המנה נזרקת ל- AX והשארית ל- DX.

קצת כאוב:
נסו לראות מה קורה אם המנה לא ניתנת לייצוג על ידי 16 סיביות (ומתי זה יקרה). האם יש לכם רעיון כיצד לפתור בעיה כאובה זו?



דוגמאות:

DIV BX
DIV Var
כאשר Var הוא משתנה זיכרון בן שני בתים.

אופרנדים מותרים:

DIV r or m

הערות:
1. פקודה זו טובה לשימוש במספרים לא מסומנים בלבד. בכדי לבצע פעולת חילוק עם מספרים מסומנים יש להשתמש בפקודה IDIV.
2. במעבד 386 יש אפשרות לבצע פעולת חילוק כאשר op הוא אופרנד בן 32 ביט. יהיה תרגיל נחמד לבדוק לאן יזרקו את השארית והמנה ומיהו המחולק (יש ניחושים ?).

מפתח סימונים:

r	AL,BL,CL,DL,AH,BH,CH,DH /AX,BX,CX,DX,SI,DI,BP,SP/ 386 – EAX,EBX,ECX,EDX,ESI,EDI,EBP,ESP
imm	קבועים מספריים (שניתן לייצגם ע"י 8,16,32 ביט)
m	ערך היושב בכתובת מסוימת בזיכרון (עד גודל 32 ביט)

פעולות אריתמטיות עם נשא

קיימות באסמבלר פעולות הנעזרות בסיבית הנשא של אוגר הדגלים - **CF**.

Add with Carry - ADC

ADC OP1 , OP2 פקודה זו מחברת את ***OP2*** ל- ***OP1*** אך עם התחשבות בנשא, כלומר , מוסיפים את הערך היושב בדגל הנשא לתוצאה (ברור שאם דגל הנשא כבוי תתפקד הפקודה כמו ***ADD*** רגיל) .

דוגמא: חיבור השרשור של Hnum ו- Lnum לאוגר DX כאשר Hnum משתנה בן 8 סיביות ומשמש בתור החלק היותר משמעותי בשרשור ו- Lnum גם הוא משתנה בגודל בית אך משמש בתור החלק הפחות משמעותי בשרשור. מחברים את Lnum ל- DL ע"י ***ADD*** רגיל ובכדי לחבר את 2 החלקים היותר משמעותיים צריך להתחשב בעובדה , שאולי בעת החיבור הקודם היתה חריגה מ- 8 הסיביות המוקצבות והתקבל נשא, לכן משתמשים בפקודה ***ADC***.
ואז, אם התקבל נשא בחיבור שני החלקים הפחות משמעותיים הוא לא יאבד מכיוון שמוסיפים אותו לתוצאת חיבור שני החלקים המשמעותיים יותר. ברור שאת שורות קוד אלו ניתן לבצע ביתר קלות באופן הבא : העברת הערך Lnum למשל לאוגר AL, והעברת הערך Hnum ל- AH וביצוע הפקודה : ADD DX, AX.

```
.DATA
Lnum DB 80h
Hnum DB 20h
.CODE
:
MOV DX,3090h
ADD DL,Lnum
ADC DH,Hnum
```

Subtraction with Borrow – SBB

SBB OP1 , OP2 פקודה זו מחסירה את ***OP2*** מ- ***OP1*** ובנוסף מחסירה את הערך היושב ב - CF מהתוצאה (ברור שאם דגל הנשא כבוי תתפקד הפקודה כמו ***SUB*** רגיל) .
לתזכורת, דגל הנשא נדלק גם כאשר מתבצעת פעולת חיסור המלווה ב - ***Borrow*** ולכן ה - CF משמעותי כאן. גם לפקודה זו יש תפקיד חשוב בפעולה על אופרנדים משורשרים.

דוגמא: כידוע, לא ניתן לבצע פעולת חיסור בין שני אופרנדים של זיכרון. ולכן פעולת השרשור במקרה זה היא בלתי נמנעת בהנחה ואנו משתמשים בשרותי מעבד 8086 בלבד. מתבצעת כאן העברה של הערך הנמצא ב- x אל האוגרים המשורשרים : DX:AX ואחר כך חיסור AX מהחלק הפחות משמעותי של y וחיסור DX מהחלק המשמעותי יותר של y תוך התחשבות מלאה בהלוואה שהתרחשה (אם התרחשה) בחיסור החלקים הפחות משמעותיים.
לאחר ביצוע שורות קוד אלו נקבל ש : **y = y – x**.

```
.DATA
x DD 50976549h
y DD 70903798h
.CODE
:
MOV AX, WORD PTR x
MOV DX,WORD PTR x+2
SUB WORD PTR y, AX
SBB WORD PTR y+2, DX
```