$t_1 = b + c$

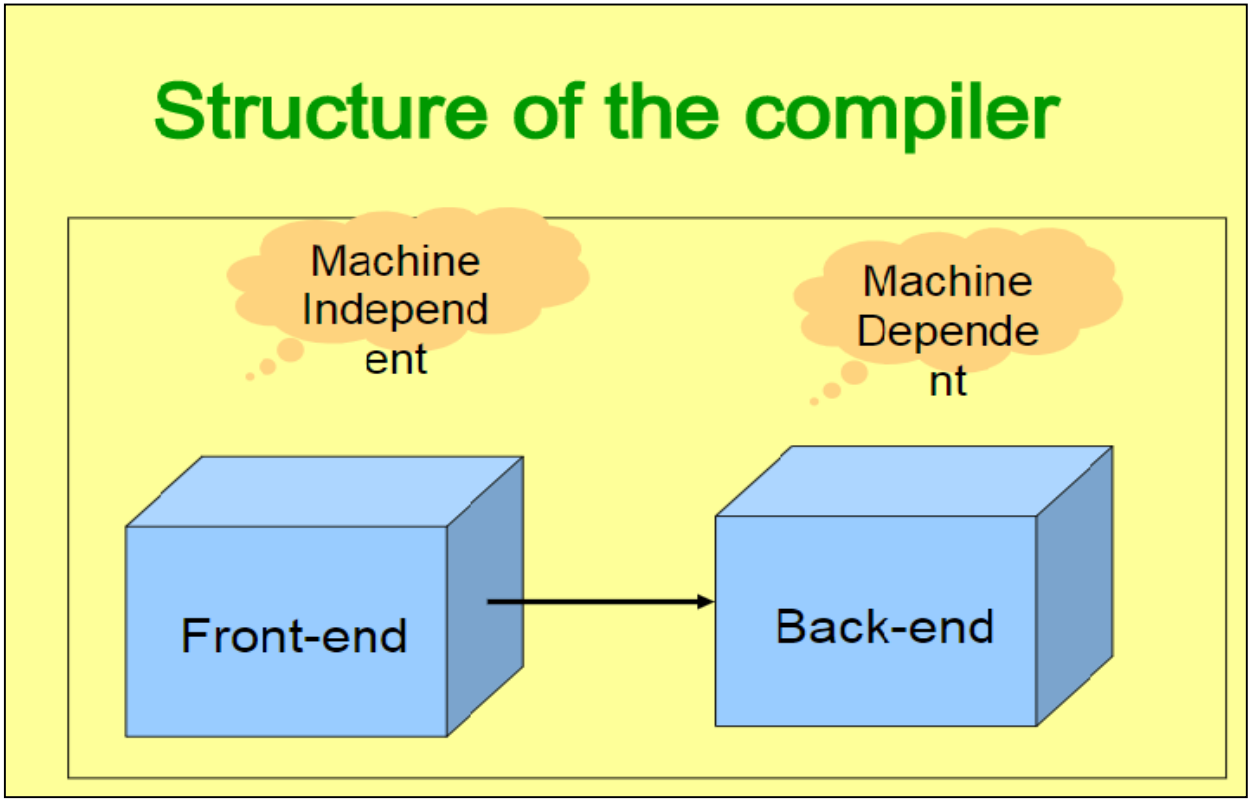
$t_2 = \text{uminus } t_1$

$t_3 = a * t_2$

Three address code

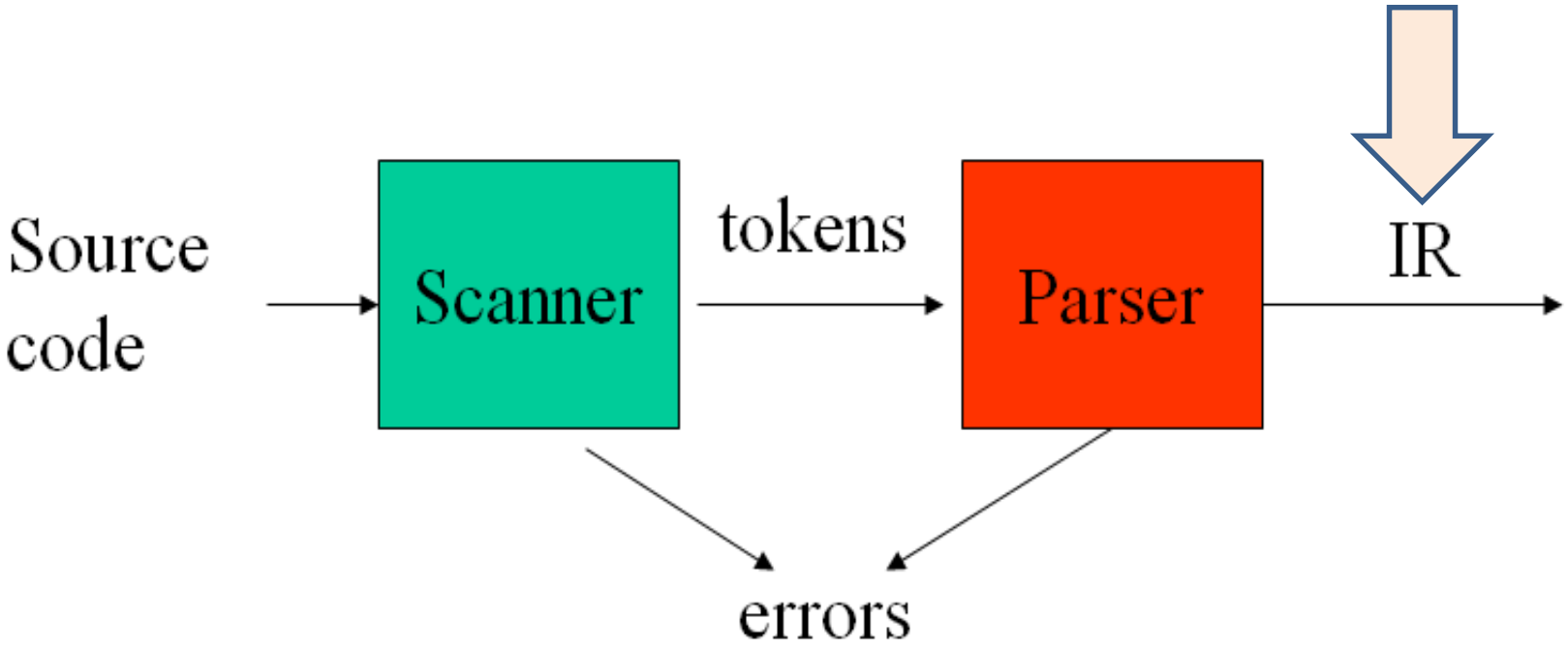
נחלק את שלבי ההידור ל 2- חלקים: **back end** ו **front end-**

ה **front - end -** כולל שלבים התלויים בשפת המקור ולא תלויים בשפת היעד.
ה – **back - end** מייצר קוד של שפת המטרה.



Three address code

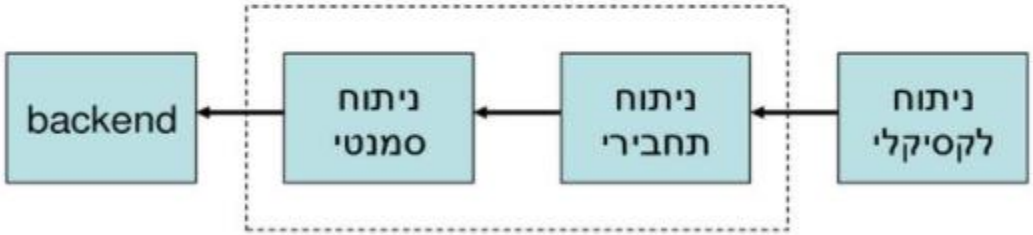
Intermediate **R**epresentation (**IR**)



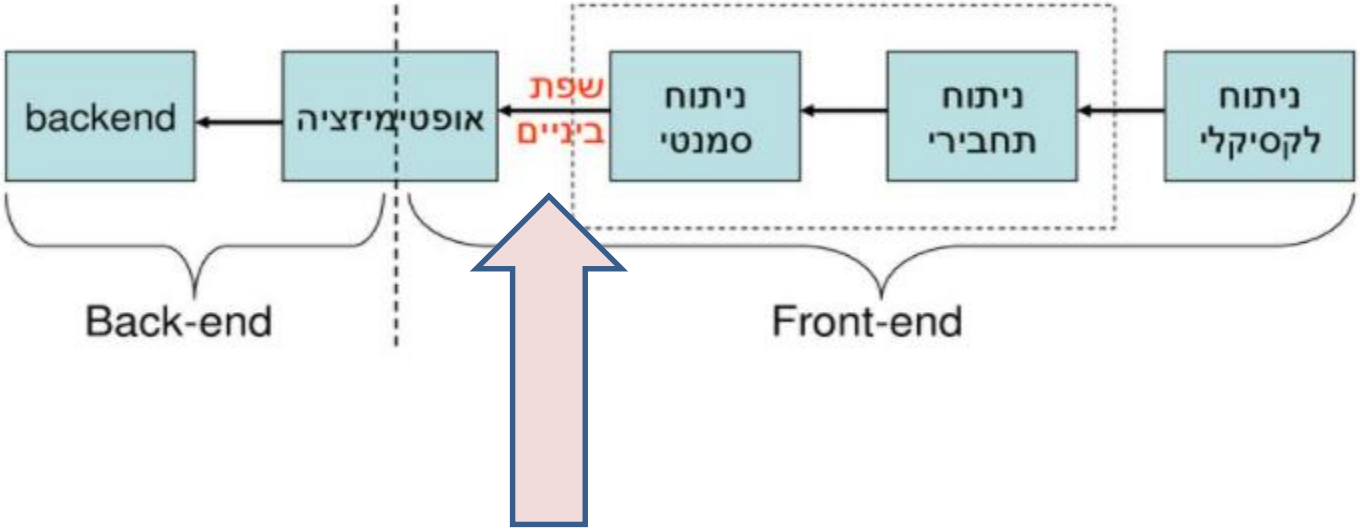
An **I**ntermediate **R**epresentation is a representation of a program “between” the source and target languages.

Three address code

- עד כה ראינו:



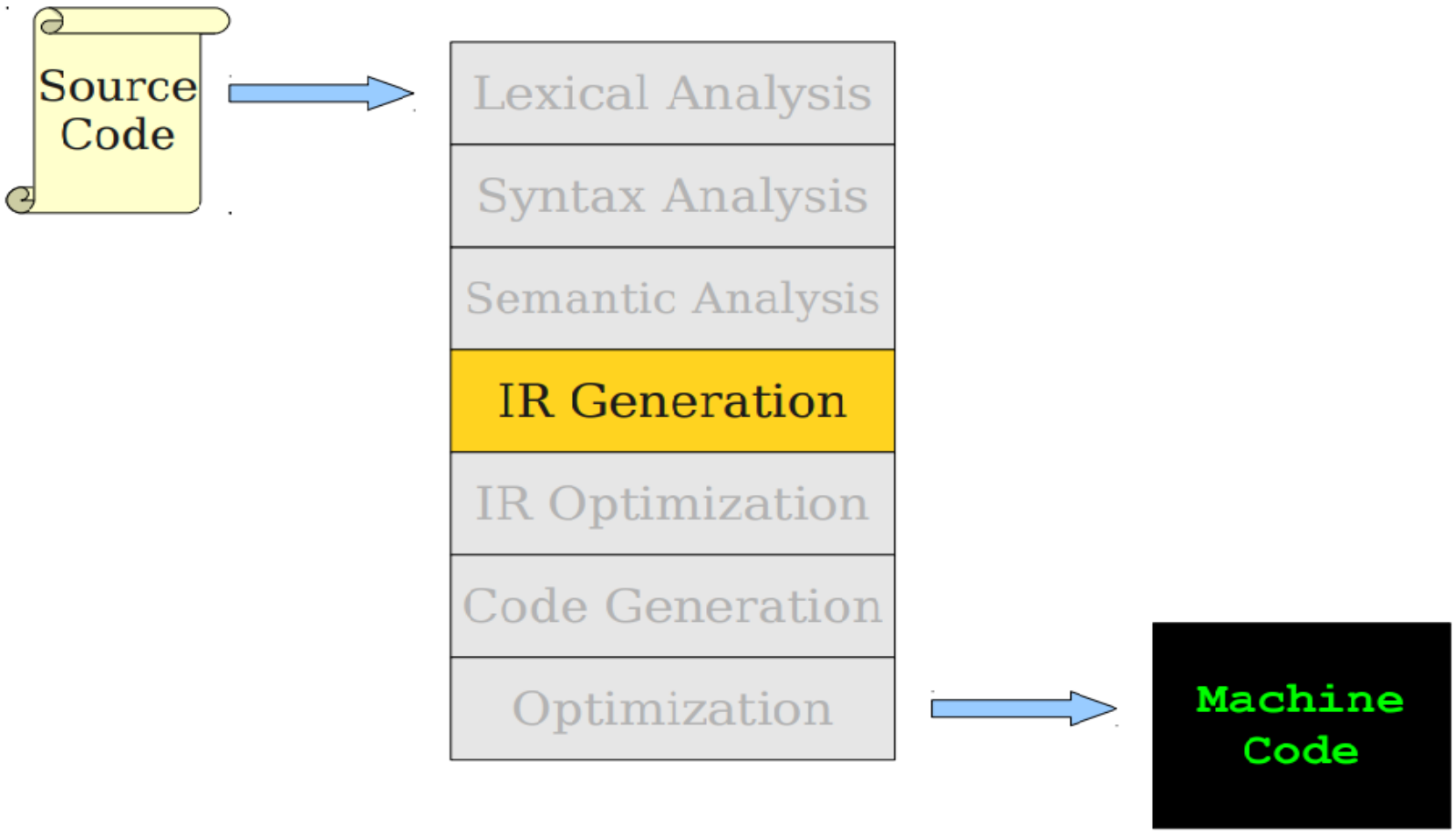
- בפועל:



שפת הביניים / שפת הרביעיות

Three address code

Where We Are



Three address code

- בשלב התרגום לקוד הביניים אנחנו מתרגמים את הקוד לשפה שאינה שפת היעד, בדרך לביצוע אופטימיזציות נוספות.

מדוע לא לתרגם ישר לשפת היעד ?

א. תרגום לשפת ביניים מאפשר פיתוח מהיר יותר של קומפיילר למערכת חדשה. נדרש במקרה כזה לכתוב רק ה- Back-end בשפת אסמבלר של מעבד מסוים .

```
CODE SEGMENT
     ASSUME CS:CODE,DS:DATA
START:
          MOV AX,DATA
          MOV DS,AX

          LEA DX,STR
          MOV AH,9
          INT 21H

          MOV AH,4CH
          INT 21H
CODE ENDS

END START
```

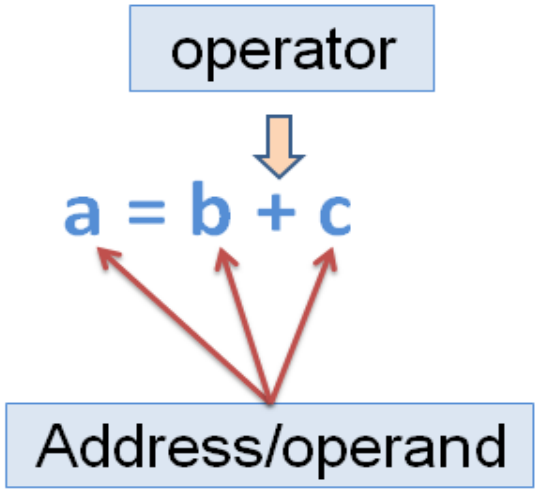
ב. פיתוח אופטימיזציות כלליות הפועלות על קוד הביניים.

Three address code

- שפת הביניים אותה נציג: **שפת הרביעיות**.
  זוהי שפת ביניים *דמוית שפת אסמבלר* בה כל פקודה מורכבת לכל היותר מארבעה אלמנטים:

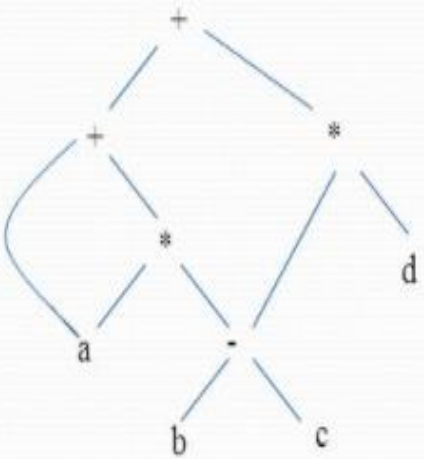
  אופרטור (הפעולה) **ושלוש כתובות** (האופרנדים).

- **קוד 3 כתובות / 3AK / TAK / Three address code**
  הוא שם נוסף מקובל לשפה זו.



Three address code

- In a three address code there is at most one operator at the right side of an instruction
- Example:



t1 = b – c
t2 = a * t1
t3 = a + t2
t4 = t1 * d
t5 = t3 + t4

a + a*(b-c) + d*(b-c)

Example : x + y * z

t1 = y * z

t2 = x + t1

Three address code

Types of Three-Address Code

- Assignment statement x := y op z
- Assignment statement x := op y
- Copy statement x := y
- Unconditional jump goto L
- Conditional jump if x relop y goto L

Three address code

- ההגדרה המלאה של שפת הרביעיות מכילה פקודות רבות :
  פקודות לעבודה נוחה עם פונקציות, פקודות לעבודה עם מערכים וכו'.
  לצורך פשטות ההסבר נדגים את פקודות הבאות בלבד.

פעולה אריתמטית	t1 := t2 + t3
קפיצה לא-מותנית Label זהו כתובת קבועה.	goto label
קפיצה מותנית relop זהו אחד מאופרטורי ההשוואה (<=, ==, ...) Label זהו כתובת קבועה	if t1 relop t2 goto label
תווית ניתן להצמיד יותר מתווית אחת לאותה הפקודה.	Label:
משפט העתקה	x := y

Three address code

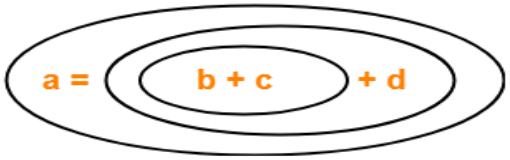
דוגמה 1 :

הביטוי הבא:

$$a = b + c + d$$

ייוצג באופן הבא בשפת הביניים:

$$t_1 := b + c$$
$$t_2 = t_1 + d$$
$$a := t_2$$



טעויות נפוצות בהבנת התחביר של השפה:

- בשפת ביניים אין else. לא קיים ביטוי ... else ... if
- התנאי בקפיצה אינו מורכב – לא ניתן לעשות && או || בתנאי ה-if.
- ניתן לקפוץ רק לכתובות קבועות. goto x כאשר x משתנה זו אינה פקודה חוקית.

Three address code

דוגמה 2 :

A popular form of intermediate code used in optimizing compilers is three-address statements.

Source statement:

$$x = a + b * c + d$$

Three address statements with temporaries t_1 and t_2:

$$t_1 = b * c$$
$$t_2 = a + t_1$$
$$x = t_2 + d$$

Three address code

דוגמה 3 :

```
int a;
int b;

a = 5 + 2 * b;
```

TAC allows for instructions with two operands.

```
_t0 = 5;
_t1 = 2 * b;
a = _t0 + _t1;
```

Three address code

דוגמה 4 :

```
while( s < d )
        if( x > y )
                z = a + b * c;
        else
                z = 0;
```

```
      t0 = s < d
L1:   if  t0  goto  L2
      goto L4
L2:   t2 = x > y
      if  t2  goto  L3
      z = 0
      goto  L1
L3:   t3 = b * c
      t4 = t3 + a
      z = t4
      goto  L1
 L4:
```

Three address code

דוגמה 4 (הסבר) :

Jump Statements

source statement like if–then–else and while–do cause jump in the control flow through three address code so any statement in three address code can be given label to make it the target of a jump.

The statement

goto L ⇦ **This is a unconditional jump**

Cause an unconditional jump to the statement with label L. the statement

Three address code

דוגמה 4 (הסבר) :

if x relop y goto L ⇦ **This is a conditional jump**

Causes a jump to L condition if and only if

Boolean condition is true.

This instruction applies relational operator relop (>,=,<, etc.)

to x and y, and executes statement L next of x statement x relop y. If not, the three address statement following if x relop y goto L is executed next, as in the usual sequence.

Three address code

תרגיל :

TEXT file:(gen3.t)

```
if(a1>22) then
                f2=a3;
else
                e3=b52;
```

```
  t0 : a1 > 22
  t1 : not t0
  if  t1 goto  L1
  t2 : f2 = a3
  goto L2
L1:
    t3 : e3 = b52
L2:
```

Three address code

LEX file:(gen3.l)

```
%%
if  return IF;
then  return THEN;
else  return  ELSE;
[a-z]+[a-z0-9]*   return ID;
[0-9]+ { yylval = atoi(yytext); return NUM; }
[ \t] printf(" ");
.  return yytext[0];
.|\n  ;
%%
```

Three address code

YACC file:(gen3.y)

```
%{
  #include <string.h>
  char st[10][10];
  int top=0;
  char i_l[2]="0";
  char temp[2] = "t";
  int label[20];
  int lnum=0,ltop=0;
%}
%token IF,THEN,ELSE,ID,NUM
%%
S: IF '(' E ')' { if_f(); } THEN E ';' { then_f(); } ELSE E ';' { else_f(); };
E: V C { push(); }  E { codegen(); }
  | V
  | NUM { push(); };
C: '=' | '>' | '<' ;
V: ID { push(); }
%%
```

Three address code

```
#include "lex.yy.c"
main()
{
   return yyparse();
}
int yyerror()
{
    printf("ERROR\n");
    return 0;
}
push()
{
     strcpy(st[++top], yytext);
}
```

Three address code

```
codegen()
{
   strcpy(temp,"t");
   strcat(temp,i_l);
   printf("%s : %s %s %s \n",temp,st[top-2],st[top-1],st[top]);
   top-=2;
   strcpy(st[top],temp);
   i_l[0]++;
 }

if_f()
{
   lnum++;
   strcpy(temp,"t");
   strcat(temp,i_l);
   printf("%s : not %s\n",temp,st[top]);
   printf("if %s goto L%d\n",temp,lnum);
   i_l[0]++;
   label[++ltop]=lnum;
}
```

Three address code

```
then_f()
{
   int x;
   Inum++;
   x=label[Itop--];
   printf("goto L%d\n",Inum);
   printf("L%d:\n",x);
   label[++Itop]=Inum;
}


else_f()
{
   int y;
   y=label[Itop--];
   printf("L%d:\n",y);
}
```

Three address code

nano gen3.t	בניית קובץ טקסט
nano gen3.l	בניית קובץ LEX
nano gen3.y	בניית קובץ YACC
lex gen3.l	קומפילציה LEX
yacc gen3.y	קומפילציה YACC
cc -o gen3 y.tab.c –ll –Ly	קומפילציה C
./gen3<gen3.t	הרצת הקובץ text2