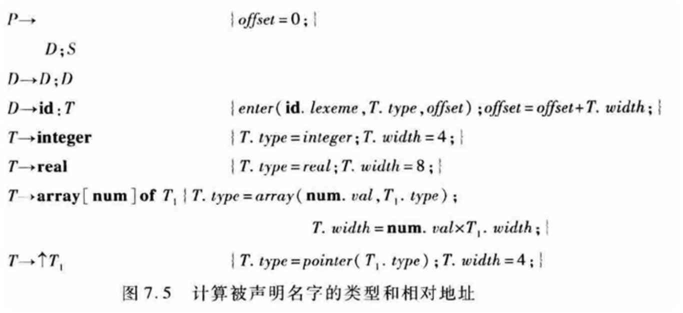
Homework6.

7.2 (c).
```
define dso_local void @main() {
        t1 = alloca i32.
        t2 = alloca [10x i32]
         br label LOOP
  LOOP:
        t3 = load i32, i32* t1
        t4 = icmp slt i32 t3 10
        br i1 t4, label TRUE, lable FALSE
  TRUE:
        t5 = getelementptr [10xi32], [10xi32]* t2, i32 0, i32 t3
        store i32 0, i32* t5
        br label LOOP
  FALSE:
         ret void.
 }
```

7.5.



$P\to$	{ *offset* = 0; }
$D;S$	
$D\to D;D$	
$D\to$ **id** : T	{ *enter*(**id**. *lexeme*, *T. type*, *offset*) ; *offset* = *offset* + *T. width*; }
$T\to$ **integer**	{ *T. type* = *integer*; *T. width* = 4; }
$T\to$ **real**	{ *T. type* = *real*; *T. width* = 8; }
$T\to$ **array**[**num**] **of** T_1	{ *T. type* = *array*(**num**. *val*, T_1. *type*) ; *T. width* = **num**. *val*×T_1. *width*; }
$T\to\uparrow T_1$	{ *T. type* = *pointer*(T_1. *type*) ; *T. width* = 4; }

图 7.5 计算被声明名字的类型和相对地址

设继承属性为 i

$P \to \{ P.i = D.i = 0 \}\ D ; S$

$D \to \{ D_1.i = D.i \}\ D_1 ; \{ D_2.i = D_1.i \}\ D_2.$

$D \to id ; T \{ enter(id.lexeme, T.type, D.i) ; D.i = D.i + T.width)$

$T \to integer\ \{ T.type = integer ; T.width = 4 \}$

$T \to real\ \{ T.type = real ; T.width = 8 \}$

$T \to array[num]\ of\ T_1\ \{ T.type = array(num.val, T_1.type)$
$T.width = num.val \times T_1.width \}$

$T \to \uparrow T_1\ \{ T.type = pointer(T_1.type), T.width = 4 ; \}$

7.12.
```
t1 = x × 5
t2 = t1 + y
t3 = t2 × 4
t4 = Base(A)
t5 = t4 + t3
Store i32 z , i32* t5
```

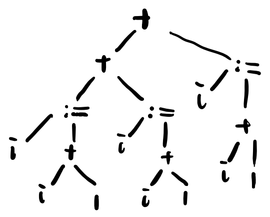
8.1 e.

```
MOV  e   R0
ADD  f   R0
MUL  d   R0
MOV  b   R1
ADD  c   R1
MOV  a   R2
DIV  R1  R2
SUB  R0  R2
```

8.2 e.

```
MOV  e(Rs)  R0
MOV  f(Rs)  R1
ADD  R0     R1
MOV  d(Rs)  R0
MUL  R0     R1
MOV  b(Rs)  R0
MOV  c(Rs)  R2
ADD  R0     R2
MOV  a(Rs)  R0
DIV  R2     R0
SUB  R1     R0
```

8.6 首先 (++i)+(++i)+(++i) 的中间代码应如下



++i ⟺ i=i+1，该表达式计算结果应保留在某个寄存器中 用于上一层计算，
因此三个 i=i+1 的计算次序不会影响最终结果，为6.

对于结果为7，一定是某个 i=i+1 结果未保留在寄存器中，
gcc的处理可能产生了一条 INC 指令来完成相同的 ++i 操作

即发现了

```
MOV  0   R1   # i=0
INC  R1       # i=1
INC  R1       # i=2
MOV  R1  R2
ADD  R1  R2   # R2=4
INC  R1       # i=3
ADD  R1  R2   # output = 7.
```