UCS1602:
COMPILER DESIGN

TAC - Declarations

Session Outcomes

- At the end of this session, participants will be able to
  - Understand the concepts of intermediate code generation of declaration of variables and procedures

Outline

- Intermediate code
- Declaration of variables
- Declaration procedures

Declarations

$P \rightarrow$ { offset=0 } D

$D \rightarrow D$; D

$D \rightarrow$ **id** : T{ enter(**id**.name,T.type,offset);
offset=offset+T.width }

$T \rightarrow$ int { T.type=int; T.width=4 }

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

$T \rightarrow$ array[num] of T_1 {
T.type=array(num.val,T_1.type);
T.width=num.val*T_1.width }

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

where *enter* crates a symbol table entry with given values.

Nested Procedure Declarations

- For each procedure we should create a symbol table.

mktable(previous) – create a new symbol table where previous is the parent symbol table of this new symbol table

enter(symtable,name,type,offset) – create a new entry for a variable in the given symbol table.

enterproc(symtable,name,newsymbtable) – create a new entry for the procedure name in the symbol table of its parent.

addwidth(symtable,width) – puts the total width of all entries in the symbol table into the header of that table.

- We will have two stacks:
  - **tblptr** – to hold the pointers to the symbol tables
  - **offset** – to hold the current offsets in the symbol tables in **tblptr** stack.

Nested Procedure Declarations

P → M D { addwidth(top(tblptr),top(offset)); pop(tblptr); pop(offset) }

M → ε { t=mktable(nil); push(t,tblptr); push(0,offset) }

D → D ; D

D → proc **id** N D ; S
{ t=top(tblptr); addwidth(t,top(offset));
pop(tblptr); pop(offset);
enterproc(top(tblptr),id.name,t) }

D → **id** : T { enter(top(tblptr),id.name,T.type,top(offset));
top(offset)=top(offset)+T.width }

N → ε { t=mktable(top(tblptr)); push(t,tblptr); push(0,offset) }

Example

```
Program sort;
    var a:array[1..n] of integer
Procedure readarray
    var i:integer
Procedure exchange(i,j:integer)
   ---
  ---
Procedure quicksort(m,n:integer)
Var k,v : integer;
    function partition(x,y : integer):integer;
       var I,f : integer;
    ……
       begin(main)
   ……
   end
```

Summary

- Intermediate code
- Abstract syntax tree
- Three address code
- Implementation of TAC

Check your understanding?

1. Generate TAC for the following declaration statements

(a) int a;

real b;

(b) a: array [5] of int

b: * int (Hint: * → pointer)

(c) procedure sample

x: int;

y : real

procedure first

z:real