Compiler Assignment - 4

[Nikhat Keshari
2017A013
CL-A]

Q1 ④ SDT for type and declaration:

Interpretation

int x = 1 { x.val = 1 }

Type checking

int x = 1 { if x.val = 1 = int
assign x.val = 1
else
error
}

⑤ SDT for type checking :-

Interpretation

$E \rightarrow E_1 + E_2$ { E.val = E_1.val + E_2.val }

Type checking

$E \rightarrow E_1 + E_2$ { if E_1.type = E_2.type = int
E.type = int;
else
E.type = float;
}

⑥ SDT for switch case :-

Interpretation

switch (S)
{ case(1): S = 10;
case(2): S = 20;
default : S = 30;
}

{ if (S.val) switch
case 1 : S.val = 10;
case 2 : S.val = 20;
default : S.val = 30;
}

Type checking

```
switch (s)
{ case 1: s.val = 10;
  case 2: s = 20;
  default: s = 30;
}
```

```
{ if s.val = 1
    s.val = 10;
  elif s.val = 2
    s.val = 20;
  else
    s.val = 30;
}
```

---

q.2: a) $P \rightarrow S$

b) $S \rightarrow$ switch stmt $S \mid ":=" \mid \epsilon$

c) switchStmt $\rightarrow$ switch (id) { case Blocks }

d) caseBlock $\rightarrow$ case : stmt2 | caseBlocks | default: 2

e) ~~switch~~ stmt 2 $\rightarrow$ ":=" | Break

f) ":=" $\rightarrow$ $x = E$

g) E: Rel Ex | Add Ex | Id

(a) P.temp $\rightarrow$ new temp()
P.code $\rightarrow$ S.code || P.temp

(b) S.temp $\rightarrow$ new temp()
S.code $\rightarrow$ S.temp || ":=" || ϵ

(c) if id.code = case
then execute caseBlock

(d) caseBlock.code $\Rightarrow$ stmt2.code.
caseBlock.val $\Rightarrow$ case1.val || caseBlocks || default

(e) stmt2.val $\Rightarrow$ ":=" || Break

(f) ":=" $\Rightarrow$ x.val = E.val

(g) E $\Rightarrow$ "<" || ">" || "=" || "+" || id.code

Q.3. cgen[cond p1 ⇒ e1; p2 ⇒ e2; ... pn ⇒ en, dnoc] =

cgen(p1) // now acc ($a0) has a pointer to value of p1
lw $a0 12($a0) // read the value attr of the bool.
beq $a0 0 PRED2 // go to second predicate value is false.
cgen(e1) // now acc ($a0) has a pointer to value of e1
b DONE // evaluation of condition is complete

label PRED2:
cgen(p2) // now acc ($a0) has a pointer to value of p2
lw($a0) 12($0) // read the value attr. of the bool.
beq $a0 0 VOID // value is void if all predicates are false
cgen(e2) // now acc ($a0) has a pointer to value of e2
b DONE // evaluation of cond. is complete

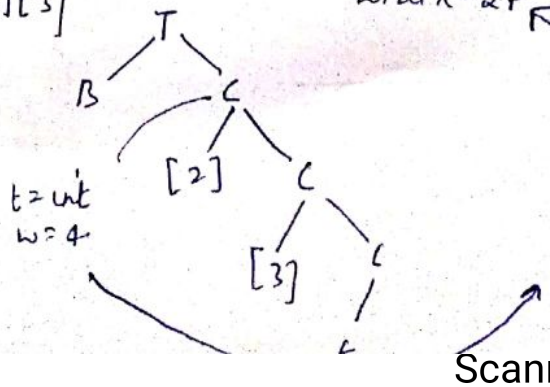
label VOID:
li $a0 0 // put VOID into the accumulator

label DONE

Q4: SDT for computing types & their width

T → BC { B.type; w = B.height }
B → int { B.type = int; B.height = 4 }
B → float { B.type = float; B.height = 8; }
C → ϵ { C.type = t; C.height = w; }
C → [num] C1 { array(num.value, C1.type);
C.width = num.value * C1.height; }

dependency graph: [2][3]



type = array(2, array(3, int))
width = 24

type = array(2, array(3, int
↑ width = 24

type = array(3, int)
↑ width = 12

type = int
width = 4

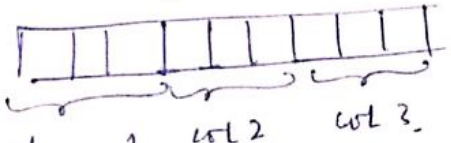
⇒ P → D { offset = 0; }

D → T id ; { T.up.put(id.lexeme, T.type, offset);
offset = offset + T.width; }

D → D1 / ε

⇒ The semantic action D → T.id ; creates a symbol table entry by executing top.put(id.lexeme, T.type, offset) here t.up denotes the current symbol table.

Array addressing :-



⇒ Location for 1D array → $\underbrace{i * width}_{\text{completed at run-time}} + \underbrace{(base_A + low * width)}_{\text{can be done at compile time}}$

⇒ The location of A[i_1, i_2] is

$base_A + ((i * low) * n_2 + i_2 - low_2) * width$.

where

$base_A$ = location of array

low_2 = index of 1st col.

low_1 = " " 1st row.

n_2 = no. of rows.

width = width of each array element can be written as

$\underbrace{((i_1 * n_2) + i_2) * width}_{\text{run-time}} + \underbrace{base_A - ((low_1 * n_2) + low_2) * width}_{\text{compile time}}$

Niket Keshari
2017A013
CS-A