4.

$S \rightarrow$ for ($\{S_1'.next = newlable\}\ S_1'$; $\{E.true = newlable.\ E.false = S.next\}\ E$;

$\{S_2'.next = newlable\}\ S_2'$) $\{S_1.next = newlable\}\ S_1$

$\{\ S.code = S_1'.code \parallel gen(S_2'.next, ":") \parallel E.code \parallel S_1.code \parallel gen(S_1.next, ":")$

$\parallel S_2'.code \parallel gen("goto", S_2'.next) \parallel gen(S.next, ":");\}$

5.

$S \rightarrow$ for (S_1' ; M_1 E ; M_2 S_2') M_3 S_1

{ backpatch (S_1'.nextlist, M_1.gotostm);

backpatch (E.truelist, M_3.gotostm);

backpatch (S_2'.nextlist, M_1.gotostm);

backpatch (S_1.nextlist, M_2.gotostm);

S.nextlist = E.falselist;

emit ("goto", M_1.gotostm);

}.

10

(a) $E \rightarrow E_1 \uparrow E_2$ {

$E_2.lable = E.lable;$

$E_2.case = not\ E.case;$

$E_1.case = false;$

if (E.case) {

$E_1.lable = E.lable;$

$E.code = E_1.code \ || \ E_2.code;$

}

else {

$E_1.lable = newlable;$

$E.code = E_1.code \ || \ E_2.code \ || \ gen(E_1.lable, ":")$

}

}.

(b) $S \rightarrow$ repeat S_1 until E

{ $S_1.next = newlable;$

$E.case = \underline{false};$

$E.lable = newlable;$

$S.code = gen(E.lable, ":") || S_1.code \ || \ gen(S_1.next, ":") \ || \ E.code$

$|| \ gen(S.next, ":");$

}

A1

$E \rightarrow$ { E_1.false = E.false , E_1.true = newlable; } E_1 ?

{ E_2.false = E.fake ; E_2.true = newlable; } E_2 :

{ E_3.true = E.false ; E_3.false = E.true } E_3

{ E.code = E_1.code || gen(E_1.true , ":") || E_2.code ||

gen (E_2.true , ":") || E_3.code ; }

A.

$E \rightarrow E_1 \; ? \; M_1 \; E_2 \; : \; M_2 \; E_3$ {

backpatch (E_1 . truelist , M_1 . gotosem);

backpatch (E_2 . truelist , M_2 . gotosem);

E. truelist = E_3 . falselist;

E. falselist = merge (E_1 . falselist . E_2 . falselist . E_3 . truelist);

}

A_3

(1) $A \rightarrow A_1 + A_2$ { A.instr = A_2.instr || A_1.instr || Plus ; }

$A \rightarrow A_1 - A_2$ { A.instr = A_2.instr || A_1.instr || Minus ; }

(2) $E \rightarrow E_1$ if B { E.instr = E_1.instr || B.instr || Cond ; }

$B \rightarrow A_1 > A_2$ { B_1.instr = Push 1 || A_2.instr || Plus || A_1.instr || Minus || Cmp ; }

$B \rightarrow B_1 \& B_2$ { B.instr = Push 1 || B_2.instr || B_1.instr || Cond || Cond ; }

$B \rightarrow !B_1$ { B.instr = Push -1 || B_1.instr || Cond || Push 1 || Add ; }

A4

(1) ① L.types = [look_up(id.entry)];

② L.types = [look_up(id.entry)] + L_1.types ;

③ R.types = [E.type];

④ R.types = [E.type] + R_1.types;

⑤ L := R

⑥ S.type = if (L.types == R.types) then ok else type_error;

(2) ① L.places = [id.place];

② L.places = [id.place] + L_1.places;

③ R.places = [newtemp]; R.code = gen (R.places[0], ":=", E.place);

④ R.places = [newtemp] + R_1.places; R.code = gen (R.places[0], ":=", E.place) || R_1.code ;

⑤ L := R

⑥ for (i = 0 to len(L.places) - 1) do

{ S.code = S.code || gen (L.places[i], ":=", R.places[i]) ; } end

(3) ① 在(2)⑥之后加入: for (i = len(R.places) to Len(L.places) - 1) do

{ S.code = S.code ||gen (L.places[i], ":=", "_ ");

if (L.types[i] == int) then

{S.eint = S.eint + [L.places[i]];}

else { S.ebool = S.ebool + [L.places[i]]; }

}

② S.code = E_1.code || E_2.code || S_1.code ;

backpatch (S.ebool , E_1.place);

backpatch (S.eint . E_2.place);