c	`dr <-newTemp()`
	`code.add(InstructionLETL(dr, c))`
	`return dr`

x	`#get the place associated to x.`
	`regval<-getTemp(x)`
	`return regval`

e_1+e_2	`t1 <- GenCodeExpr(e_1)`
	`t2 <- GenCodeExpr(e_2)`
	`dr <- newTemp()`
	`code.add(InstructionADD(dr, t1, t2))`
	`return dr`

e_1-e_2	`t1 <- GenCodeExpr(e_1)`
	`t2 <- GenCodeExpr(e_2)`
	`dr <- newTemp()`
	`code.add(InstructionSUB(dr, t1, t2))`
	`return dr`

true	`dr <-newTemp()`
	`code.add(InstructionLETL(dr, 1))`
	`return dr`

$e_1 < e_2$	`dr <- newTemp()`
	`t1 <- GenCodeExpr(e1)`
	`t2 <- GenCodeExpr(e2)`
	`endrel <- newLabel()`
	`code.add(InstructionLET(dr, 0))`
	`#if t1>=t2 jump to endrel`
	`code.add(InstructionCondJUMP(endrel, t1, ">=" , t2)`
	`code.add(InstructionLET(dr, 1))`
	`code.addLabel(endrel)`
	`return dr`

FIGURE 1 – 3@ Code generation for numerical or Boolean expressions (t1 and t2 are already defined)

x = e	`dr <- GenCodeExpr(e)`
	`#a code to compute e has been generated`
	`if x has a location loc:`
	`  code.add(instructionCOPY(loc,dr))`
	`else:`
	`  storeLocation(x,dr)`

S1 ; S2	`#concat codes`
	`GenCodeSmt(S1)`
	`GenCodeSmt(S2)`

`if` b `then` $S1$ `else` $S2$	`lelse,lendif <-newLabels()`
	`t1 <- GenCodeExpr(b)`
	`#if the condition is false, jump to else`
	`code.add(InstructionCondJUMP(lelse, t1, "=", 0))`
	`GenCodeSmt(S1) #then`
	`code.add(InstructionJUMP(lendif))`
	`code.addLabel(lelse)`
	`GenCodeSmt(S2) #else`
	`code.addLabel(lendif)`

`while` b `do` S `done`	`ltest,lendwhile <-newLabels()`
	`code.addLabel(ltest)`
	`t1 <- GenCodeExpr(b)`
	`code.add(InstructionCondJUMP(lendwhile, t1, "=", 0))`
	`GenCodeSmt(S)  #execute S`
	`code.add(InstructionJUMP(ltest))  #and jump to the test`
	`code.addLabel(lendwhile)  #else it is done.`

FIGURE 2 – 3@ Code generation for Statements