**Practical No. 06**

**Name: Amarsingh Kashyap**

**Roll No: 101**

**Branch: CSE 2nd Shift**

**Aim:** Write a program to generate three address code for the given language construct using SDTS.

* **while loop with if then else**

**CODE :**

**from** Block **import** Block  
  
temp = 0  
E = **None**IFs = []  
ELSEs = []  
  
  
**def** readinput(i):  
 **global** E  
 **global** IFs  
 **global** ELSEs  
 **global** d  
 E = **None** IFs = []  
 ELSEs = []  
 d.clear()  
  
 file = open(**"input"**+str(i)+**".txt"**, **"r"**)  
 **return** file  
  
  
**def** process(STMTS, code):  
 **global** temp  
  
 **if** code == 1:  
 **for** s1 **in** STMTS:  
 temp = temp + 1  
 address = **"t"**+str(temp)  
 s1 = s1.split(**"="**)  
 tempStorage = address+**" = "**+s1[1].strip()  
 assignment = s1[0].strip()+**" = "**+address  
 IFs.append(tempStorage)  
 IFs.append(assignment)  
  
 **else**:  
 **for** s2 **in** ELSE:  
 temp = temp + 1  
 address = **"t"**+str(temp)  
 s2 = s2.split(**"="**)  
 tempStorage = address+**" = "**+s2[1].strip()  
 assignment = s2[0].strip()+**" = "**+address  
 ELSEs.append(tempStorage)  
 ELSEs.append(assignment)  
  
d = dict()  
  
**def** getTAC(E, S1, S2):  
 print(**"\n==========================================================="**)  
 start = int(input(**"Base Address: "**))  
 print(**"==========================================================="**)  
 print()  
 address = start  
  
 entry = E.code+**" goto "**+str(start + 2)  
 E.next = start  
  
 M1 = Block(**None**, **None**, **None**)  
 M1.next = start+2  
 TAC.append(entry)  
  
 TAC.append(**"goto "**+str(start + len(S1.code) + 3))  
  
 M2 = Block(**None**,**None**, **None**)  
 M2.next = start+len(S1.code)+3  
  
 S1.next = str(start + len(S1.code) + 3 - 1)  
  
 TAC.extend(E.true.code)  
  
 TAC.append(**"goto"**+**" "**+str(start +(len(IFs)+len(ELSEs)+3)))  
  
 S2.next = str(start +(len(IFs)+len(ELSEs)+3))  
  
 TAC.extend(E.false.code)  
 TAC.append(**"END"**)  
  
 semantics = []  
 semantics.append(**"Backpatch("** + str(E.next) + **","** + str(M1.next) + **")"**)  
 semantics.append(**"Backpatch("** + str(E.next) + **","** + str(M2.next) + **")"**)  
 semantics.append(**"S.next = merge("** + str(S1.next) + **","** + str(S2.next) + **")"**)  
 **return** address, semantics  
  
  
**if** \_\_name\_\_ == **"\_\_main\_\_"**:  
 EI = 1  
 print(**"\n==========================================================="**)  
 **for** i **in** range(EI):  
 print(**"Given Input :\n==========================================================="**)  
 code = readinput(i+1)  
 program = code.read()  
 print(program)  
  
 LOC = program.split(**"\n"**)  
 TAC = []  
  
 **for** line **in** LOC:  
 **if "if" in** line:  
 E = line  
 IF = []  
 **if "else" in** LOC:  
 till = LOC.index(**"else"**)  
 **else**:  
 till = len(LOC)-1  
 **for** stmt **in** LOC[LOC.index(line)+1:till]:  
 IF.append(stmt.strip())  
 process(IF, 1)  
  
 **elif "else" in** line:  
 ELSE = []  
 **for** stmt **in** LOC[LOC.index(line)+1:]:  
 ELSE.append(stmt.strip())  
 process(ELSE, **None**)  
  
  
 S1 = Block(IFs, **None**, **None**)  
 S2 = Block(ELSEs, **None**, **None**)  
 E = Block(E, S1, S2)  
  
  
 sementics = []  
 add, sementics = getTAC(E, S1, S2)  
  
 **for** each **in** TAC:  
 print(add, **":"**, each)  
 d[add] = each  
 add = add+1  
 print(**"\n==========================================================="**)  
 print(**"Sementics Analysis:\n----------------------------------"**)  
 **for** s **in** sementics:  
 print(s)  
  
 print(**"===========================================================\n"**)

* **BLOCK.py**

**class** Block:  
 **def** \_\_init\_\_(self, code, true, false):  
 self.code = code  
 **if** true **is not None and** false **is not None**:  
 self.true = true  
 self.false = false  
 self.next = **None**

**OUTPUT :**





