```
import tkinter as tk
import winsound
from tkinter import *
from tkvideo import tkvideo
from
tkinter import messagebox

operator = ['+','-','*','/','@']
def operand(exp):

r=exp[::-1]
  for j in range(len(r)):
    if r[j] not in operator:
       return j
def
generate_assembly_code(exp):
    stack = []
    temp = 1
    assembly = ""

operator = ['+','-','*','/','@']
    l = operand(exp)
    n=len(exp)
    for i in
range(n):
        char = exp[i]
        if char.isalpha():
            stack.append(char)

      elif char == "@":
                op = stack.pop()
                assembly
+= f"N\n"
                if temp%2 != 0:
                   assembly += f"ST
${str(temp%2)}\n"
                stack.append(f"${str(temp%2)}")

   temp += 1
        elif ((exp[i-1]) in operator and (exp[i]) == '-'):
                op =
stack.pop()
                if op[0] != "$":
                   assembly +=
f"L {op}\n"
                assembly += "N\n"
                if temp%2
!= 0:
                   assembly += f"A ${str(temp%2)}\n"

stack.append(f"${str(temp%2)}")
                temp += 1

else:
            operand2 = stack.pop()
            if len(stack) != 0:

operand1 = stack.pop()

            if(exp[i] in operator and exp[i-1] not in
operator):
              if operand1[0] != '$':
                 assembly += f"L
{operand1}\n"
              if char == '+':
                assembly += f"A
{operand2}\n"
              elif char == '-':
                assembly += f"S
{operand2}\n"
              elif char == '*':
                assembly += f"M
{operand2}\n"
              elif char == '/':
```

```
                assembly += f"D
{operand2}\n"

              if temp < 3:
                if(exp[i] in operator and
exp[i-1] not in operator):
                  if exp[i+1] != "@":

  assembly += f"ST ${str(temp)}\n"
                  if i != (len(exp)-1):

           if(exp[i+1]) == "+":
                      assembly += f"A
${str((temp%2)+1)}\n"
                    if(exp[i+1]) == "*":

       assembly += f"M ${str((temp%2)+1)}\n"
                    if(exp[i+1]) ==
"/":
                       assembly += f"D ${str((temp%2)+1)}\n"


              else:
                if(exp[i] in operator and exp[i-1] not in
operator):
                  if(exp[i+1]) == "+":
                    assembly +=
f"A ${str((temp%2)+1)}\n"
                  if(exp[i+1]) == "*":

           assembly += f"M ${str((temp%2)+1)}\n"
                  if(exp[i+1]) ==
"/":
                    assembly += f"D ${str((temp%2)+1)}\n"


                if(i < n-l and exp[i] in operator and exp[i-1] not in operator):

               assembly += f"ST ${str((temp%2)+1)}\n"


stack.append("$" + str(temp%2))
            temp += 1
    x = stack.pop()

 if x != '$0':
      if assembly[-2] != '1':
         if exp[-1] == '/':

assembly += 'D ' + x
         elif exp[-1] == '*':
            assembly += 'M ' + x

elif exp[-1] == '+':
            assembly += "A " + x
    return assembly

def
play_button_sound(sound_file_path):
    winsound.PlaySound(sound_file_path,
winsound.SND_FILENAME)

def exit_video():
        window.destroy()

play_button_sound("button_sound.wav")

def create_window():
   root =
tk.Toplevel()
   root.title("Code Generation")

   img = tk.PhotoImage(file =
"background.png")
   bg = tk.Label(root, image = img)
```

```
   bg.image = img

bg.pack()
   root.attributes("-fullscreen", True)

   label = tk.Label(root,
text="Enter Postfix Expression", font = ("Aerial", 15, "bold"
))
   label.place(x = 130, y = 75)

   entry = tk.Entry(root, font=("Times New
Roman", 15),width=30)
   entry.place(x = 100, y = 130)

   def translate():

postfix_expression = entry.get()
    try:
       assembly_code =
generate_assembly_code(postfix_expression)
    except:

messagebox.showerror("Error", "Invalid input! Please enter a valid Postfix
Expression.")
    output.delete(1.0, tk.END)

output.tag_configure("center", justify='center')
    output.insert("1.0",
assembly_code)
    output.tag_add("center", 1.0, "end")

play_button_sound("button_sound.wav")

   translate_button = tk.Button(root,
text="Translate",font = ("Aerial", 15), fg = 'blue', command=translate)

translate_button.place(x= 200 ,y = 165)

   T = Text(root, height = 11, width = 55 ,font =
("Aerial",15, "bold"), bg = 'pink')
   T.place(x = 600, y = 250)

   l =
Label(root, text = "Commands", font = ("Times New Roman", 25,
"bold"))
   l.place(x = 800, y = 150)

   detail = '''
  L : Load the operand
into the register
  A : Add the operand to the contents of the register
  S : Subtract the
operand from the contents of the register
  M : Multiply the contents of the register by the
operand
  D : Divide the contents of the register by the operand
  N : Negate the contents of
the register
  ST: Store the contents of the register in the operand location
  $n: Temporary
storage locations
  n : A Single Digit
  '''
   T.insert(tk.END, detail)

   exit =
Button(root, text="Exit", width = 5,command=exit_video)
   exit.place(relx=0.95,
rely=0.05, anchor=CENTER)

   output_label = tk.Label(root, text="Assembly Code",
font = ("Aerial", 20, "bold"))
   output_label.place(x = 145, y = 230)
```

```
   output = tk.Text(root, width=30, height=20, font = ('Cooper black' , 12), fg =
'black',borderwidth = 1, relief = 'solid')
   output.place(x = 100, y = 275)

def
play_video():
    window.title("CODE GENERATION")

window.attributes("-fullscreen", True)

    Video = Label(window)

Video.pack(fill=BOTH, expand=YES)

    player = tkvideo("video.mp4", Video, loop=1,
size=(window.winfo_screenwidth(), window.winfo_screenheight()))

    def button_click():

   create_window()
       play_button_sound("button_sound.wav")

    btn =
tk.Button(window, text = "Proceed", height = 1, width = 8,font = ("Aerial",
20, "bold"), fg ='blue', borderwidth = 3, relief = 'solid', command=button_click)

 btn.place(relx=0.5, rely=0.9, anchor=CENTER)

    exit_button = Button(window,
text="Exit",width = 5, command=exit_video)
    exit_button.place(relx=0.95,
rely=0.05, anchor=CENTER)

    player.play()

window = Tk()
app =
play_video()
window.mainloop()
```