JC-62 Simulator

COA PROJECT

-Kushagra Nigam

Features & Working

JC-62 simulator help us to perform all the instructions of JC-62 machine. JC-62 is a simple computer which helps us to perform simple addition, subtraction and branching. A single 12-bit-wide bus provides for exchange of information between pairs of registers within the data path section .The registers and the 256 X 12 bit RAM memory are controlled by 16 control signals . Most of the registers have Load (L) and Enable (El signals . An active L signal to a register causes the contents of the bus to be clocked into that register on the next rising pulse from the system clock . An active E signal enables the tristate outputs of the register, thereby making its contents available to the bus .

Processing of data is done by the Arithmetic-Logic-Unit (ALU), a circuit that is capable of adding or subtracting the 12-bit numbers contained in its two input registers : the accumulator (ACC) and register B . The operation performed by the ALU is selected by the Add (A) or Subtract(S) control signals The accumulator also contains a single flip-flop that is set whenever its contents are negative (i .e . ,whenever the leading bit is set--meaning a negative two's complement number) . The value of this "negative flag" provides input to the controller/ sequencer, and, as we shall see, permits implementation of conditional branching instructions .

The machine's RAM memory is accessed by first placing the 8-bit address in the Memory Address Register (MAR). An active Read (R) control signal to the RAM will then cause the selected word from the RAM to appear in the Memory Data Register(MDR) . An active Write (W) signal, on the other hand, will cause the word contained in the MDR to be stored in the RAM at the address specified by the MAR .

This JC-62 simulator helps us to perform all types of code that can run through JC-62 machine.

Front-end

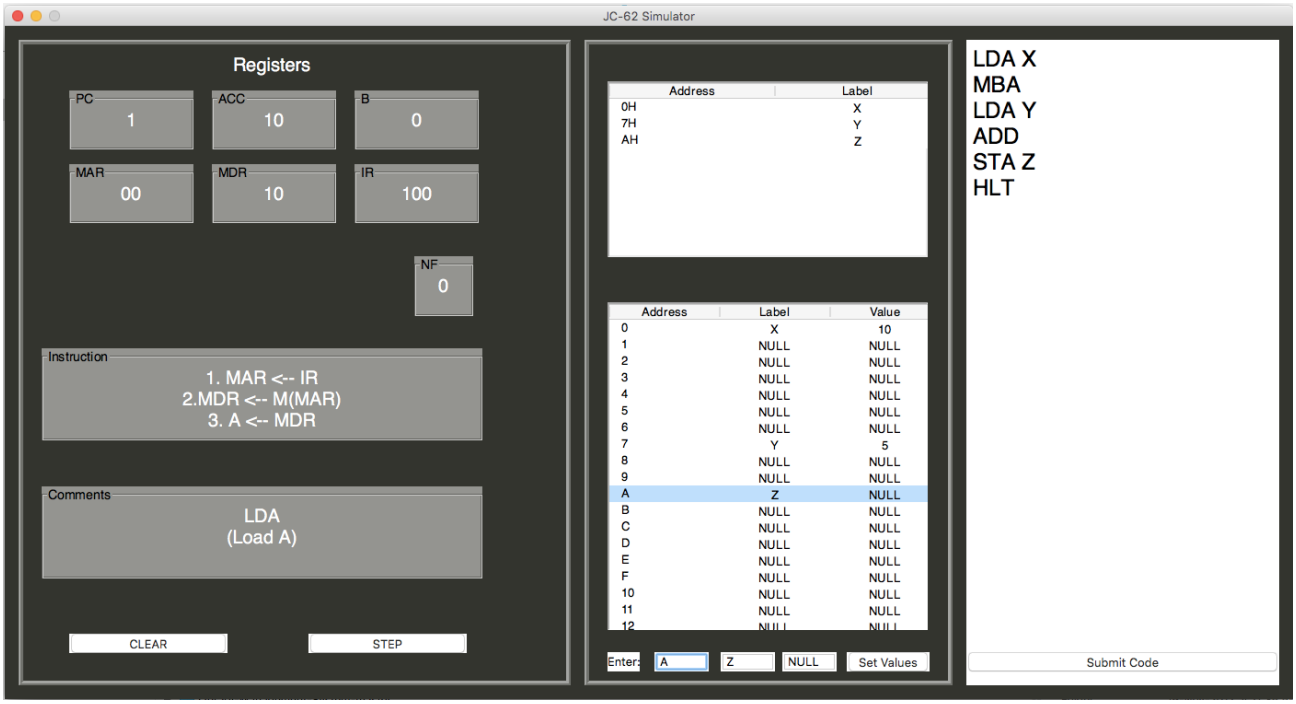
We have designed a GUI of JC-62 which contains all the registers, flags, RAM, text area. The designing of this GUI is done through Python.

Registers and components:

1. Program Counter
2. MAR
3. MDR
4. ALU
5. ACCUMULATOR
6. B Register
7. IR

Flags:

1. Negative Flag



Back-End

The back-end is also done through python. And connection between the GUI and back-end is also done through Python.

```
    global allIns
    allIns = textfield.get(1.0,END).splitlines()
    Comments.configure(text='Code Submitted')
    Instruction.configure(text='')
    print (allIns)

def EditTree(event):
    item = tree2.selection()[0]
    entry3.insert(0,tree2.item(item,"text"))
    entry2.insert(0,tree2.item(item, "value"))

def SetButton(event):
    add=int(entry3.get(),16)
    value=entry2.get()
    label=entry1.get()
    item = tree2.selection()[0]
    tree2.item(item,text=hex(add)[2:].upper(), values=(label, value))

    adInHex=str((hex(add))).upper()
    tree1.insert('','end', text=(adInHex[2:]+"H"),values=label)
def StopButton(event):
    global insCounter
    global PCCounter
    global ACCValue
    global BValue
    global NFValue
    insCounter = 0
    PCCounter = 0
    ACCValue = 0
    BValue = 0
    NFValue=0

    PC.configure(text='')
    MAR.configure(text='')
    ACC.configure(text='')
    MDR.configure(text='')
    IR.configure(text='')
    B.configure(text='')
    NF.configure(text='')
    Comments.configure(text='')
    Instruction.configure(text='')
    for i in tree2.get_children():
        tree2.delete(i)
    for i in tree1.get_children():
        tree1.delete(i)
    for i in range(0,256):
        x=str(hex(i))[2:].upper()
        tree2.insert('','end',text=x, values=("NULL", "NULL"))

"""
def InsToRAM(event):
```

Ln: 1 Col: 0

In our Back-end function of all the instruction is processed and is performed. Our program reads the instruction line by line and performs the whole program step wise. With each instruction performed PC is incremented and all the other registers also changes their value accordingly. We have also added comments which helps us to analyse our program more efficiently.

Source Code

```
from tkinter import *
import time
import tkinter.ttk as ttk
import os
from tkinter import filedialog
import tkinter as tk

#definition of global variables
allIns = []
insCounter = 0
PCCounter = 0
ACCValue = 0
BValue = 0
NFValue=0
imagesrc="fetch.png"
```

```
def retrieve(name):
    head=name.split('/')
    final=head[-1].split('.')
    return final[0]

def ControlSignals(event):
    photo=PhotoImage(file="fetch.png")
    label=tk.Label(root,image=photo,bg="black")
    label.image=photo
    label.place(x=215, y=75)


def Samples():
    ftypes = [("text files","*.txt"),('All files', '*')]
    dlg = filedialog.askopenfilename(initialdir = "sample codes",title = "Select file",filetypes = ftypes)
    fl = dlg
    if fl != '':
        textfield.delete("1.0",'end-1c')
        file_name = entry.get("1.0",'end-1c')
        file=open(fl, 'r')
        textfield.insert(END,file.read())
        root.title(retrieve(fl)+"- JC-62 Simulator")

def onOpen():
    ftypes = [("text files","*.txt"),('All files', '*')]
    dlg = filedialog.askopenfilename(initialdir = "user",title = "Select file",filetypes = ftypes)
    fl = dlg
    if fl != '':
        textfield.delete("1.0",'end-1c')
        file_name = entry.get("1.0",'end-1c')
        file=open(fl, 'r')
        textfield.insert(END,file.read())
        root.title(retrieve(fl)+"- JC-62 Simulator")

def Saveas():
    ftypes = [("text files","*.txt"),('All files', '*')]
    dlg =filedialog.asksaveasfilename(initialdir = "user",title = "Save file",filetypes = (("text
files","*.txt"),("all files","*.*")))
    fl = dlg
    if fl != '':
        file_name = entry.get("1.0",'end-1c')
        file=open(fl, 'w')
        file.write(textfield.get(1.0,END))
        file.close()
        root.title(retrieve(fl)+"- JC-62 Simulator")


def SubmitCode(event):
    global allIns
    allIns = textfield.get(1.0,END).splitlines()
    Comments.configure(text='Code Submitted')
```

```
    Instruction.configure(text='')
    print (allIns)

def EditTree(event):
    item = tree2.selection()[0]
    entry3.config(state=NORMAL)
    entry1.delete('1.0', END)
    entry2.delete('1.0', END)
    entry3.delete('1.0', END)
    entry3.insert(END,tree2.item(item,"text"))
    entry2.insert(END,tree2.item(item,"values"))
    entry3.config(state=DISABLED)

def SetButton(event):
    add=int(entry3.get("1.0",'end-1c'),16)
    value=entry2.get("1.0",'end-1c')
    label=entry1.get("1.0",'end-1c')
    item = tree2.selection()[0]
    tree2.item(item,text=hex(add)[2:].upper(), values=(label, value))
    adInHex=str((hex(add))).upper()
    tree1.insert('','end', text=(adInHex[2:]+"H"),values=label)
    entry1.delete('1.0', END)
    entry2.delete('1.0', END)
    entry3.delete('1.0', END)

def StopButton(event):
    global insCounter
    global PCCounter
    global ACCValue
    global BValue
    global NFValue
    global imagesrc
    imagesrc="fetch.JPG"
    insCounter = 0
    PCCounter = 0
    ACCValue = 0
    BValue = 0
    NFValue=0
    PC.configure(text='')
    MAR.configure(text='')
    ACC.configure(text='')
    MDR.configure(text='')
    IR.configure(text='')
    B.configure(text='')
    NF.configure(text='')
    Comments.configure(text='')
    Instruction.configure(text='')
    for i in tree2.get_children():
        tree2.delete(i)
    for i in tree1.get_children():
        tree1.delete(i)
```

```
    for i in range(0,256):
        x=str(hex(i))[2:].upper()
        tree2.insert('','end',text=x, values=("NULL", "NULL"))
def FindInRam(labeltofind):
    for x in tree2.get_children():
        if(tree2.item(x)["values"][0]==labeltofind):
            return x

def Step():
    global insCounter
    global ins
    try:
         ins = textfield.get(1.0,END).splitlines()[insCounter]
    except:
        print("Instuction out of bounds")

    print(ins)

    time.sleep(0.5)

    if(ins[:2]=='JN'):
        JN(int(ins[3:5]))
    elif(ins[:3]=='LDA'):
        LDA(ins)
    elif(ins[:3]=='STA'):
        STA(ins)
    elif(ins[:3]=='ADD'):
        ADD()
    elif(ins[:3]=='SUB'):
        SUB()
    elif(ins[:3]=='MBA'):
        MBA()
    elif(ins[:3]=='JMP'):
        JMP(int(ins[4:6]))
    elif(ins[:3]=='HLT'):
        HLT()
    insCounter+=1
def Run(event):
    i=0
    while i<len(allIns):
        Step()
        i+=1
    butt7.configure(state=DISABLED)
def InsRead(event):
    global insCounter
    try:
        ins = textfield.get(1.0,END).splitlines()[insCounter]
        print(ins)
    except:
        print("Instuction out of bounds")
```

```
    print(ins)

    time.sleep(0.5)

    if(ins[:2]=='JN'):
        JN(int(ins[3:5]))
    elif(ins[:3]=='LDA'):
        LDA(ins)
    elif(ins[:3]=='STA'):
        STA(ins)
    elif(ins[:3]=='ADD'):
        ADD()
    elif(ins[:3]=='SUB'):
        SUB()
    elif(ins[:3]=='MBA'):
        MBA()
    elif(ins[:3]=='JMP'):
        JMP(int(ins[4:6]))
    elif(ins[:3]=='HLT'):
        HLT()
    insCounter+=1
def MBA():
    global ACCValue
    global PCCounter
    global BValue
    BValue=ACCValue
    global imagesrc
    imagesrc="MBA.jpg"
    PC.configure(text=PCCounter+1)
    MAR.configure(text=PCCounter)
    ACC.configure(text=ACCValue)
    MDR.configure(text='06')
    IR.configure(text='06')
    B.configure(text=BValue)
    NF.configure(text='0')
    Comments.configure(text='MBA\n(Move A to B)')
    Instruction.configure(text='1. B <-- A')
    PCCounter=PCCounter+1
    return
def LDA(ins):
    global ACCValue
    global PCCounter
    global BValue
    global imagesrc
    imagesrc="LDA.jpg"
    treepointer = FindInRam(ins[4:])
    ACCValue = tree2.item(treepointer)['values'][1]
    AddOfLabel=tree2.item(treepointer)['text']
    if (len(str(AddOfLabel))==1):
        AddOfLabel='0'+AddOfLabel
```

```
    PC.configure(text=PCCounter+1)
    MAR.configure(text=AddOfLabel)
    ACC.configure(text=ACCValue)
    MDR.configure(text=ACCValue)
    IR.configure(text='1'+AddOfLabel)
    B.configure(text=BValue)
    NF.configure(text='0')
    Comments.configure(text='LDA\n(Load A)')
    Instruction.configure(text='1. MAR <-- IR\n2.MDR <-- M(MAR)\n3. A <-- MDR')
    PCCounter=PCCounter+1
def STA(ins):
    global ACCValue
    global PCCounter
    global imagesrc
    imagesrc="STA.jpg"
    treepointer = FindInRam(ins[4:])
    AddOfLabel=tree2.item(treepointer)['text']
    if (len(str(AddOfLabel))==1):
        AddOfLabel='0'+AddOfLabel
    PC.configure(text=PCCounter+1)
    MAR.configure(text=AddOfLabel)
    MDR.configure(text=AddOfLabel)
    IR.configure(text='1'+AddOfLabel)
    Comments.configure(text='STA\n(Store A)')
    Instruction.configure(text='1. MAR <-- IR\n2. MDR <-- A\n3. M(MAR) <-- MDR')
    tree2.item(treepointer,values=ins[4:]+" "+str(ACCValue))
    PCCounter=PCCounter+1
def ADD():
    global ACCValue
    global PCCounter
    global BValue
    global imagesrc
    imagesrc="ADD.jpg"
    PC.configure(text=PCCounter+1)
    MAR.configure(text=PCCounter)
    ACCValue=ACCValue+BValue
    ACC.configure(text=ACCValue)
    MDR.configure(text='04')
    IR.configure(text='04')
    B.configure(text=BValue)
    NF.configure(text='0')
    Comments.configure(text='ADD\n(Add B to A)')
    Instruction.configure(text='1. A <-- ALU(add)')
    PCCounter=PCCounter+1
    return
def JMP(x):
    global insCounter, PCCounter
    global imagesrc
    imagesrc="JMP.jpg"
    insCounter = PCCounter = x-1
    Comments.configure(text='JMP\n(Jump to Address)')
```

```
    Instruction.configure(text='1. PC <-- IR')
    return
def SUB():
        global ACCValue
        global PCCounter
        global BValue
        global NFValue
        global imagesrc
        imagesrc="SUB.jpg"
        PC.configure(text=PCCounter+1)
        MAR.configure(text=PCCounter)
        sub=ACCValue-BValue
        if(sub<0):
            NF.configure(text='1')
            sub=sub*-1
            NFValue=1
        else:
            NF.configure(text='0')
        ACCValue=sub
        ACC.configure(text=sub)
        MDR.configure(text='05')
        IR.configure(text='05')
        B.configure(text=BValue)
        Comments.configure(text='SUB\n(Subtract B from A)')
        Instruction.configure(text='1. A <-- ALU(sub)')
        PCCounter=PCCounter+1
        return
def JN(x):
    global insCounter, NFValue , PCCounter
    global imagesrc
    imagesrc="JMP.jpg"
    if(NFValue == 1):
        insCounter = PCCounter = x-1
    Comments.configure(text='JN\n(Jump if Negative)')
    Instruction.configure(text='1. PC <-- IR')
    return
def HLT():
    global ACCValue
    global PCCounter
    global imagesrc
    imagesrc="jc62.jpg"
    PCCounter+=1
    PC.configure(text=PCCounter)
    Comments.configure(text='HLT\n(Teriminate)')
    Instruction.configure(text=' ')
    return



root = Tk()
root.title("Untitled- JC-62 Simulator")
```

```
root.geometry("1400x731")
#root.resizable(0,0)
root.configure(background="#228b22")
frame1= Label(root, background="#191970", relief=RIDGE, borderwidth=5)
frame2= Frame(root, background="#191970", relief=RIDGE, borderwidth=5)
frame3= Frame(root,background="#191970")
label20 = Label(frame1, text="Registers", font = 'times 20', bg="#f5deb3", fg="BLACK")
label21 = Label(frame2, text="Enter:")
textfield = Text(frame3, height=700, width=300, font='times 24')

butt1 = Button(frame3, text="Submit Code" ,height=25)
butt2 = Button(frame2, text="Set Values" ,height=25)
butt3 = Button(frame1, text="STEP" ,height=25)
butt4 = Button(frame1, text="CLEAR" ,height=25)
butt5 = Button(frame1, text="Control Signals" ,height=25)
butt7 = Button(frame1, text="Run All" ,height=25)
lframe=LabelFrame(frame1,bg="#f5deb3",text="PC",font='times 14')
lframe1=LabelFrame(frame1,bg="#f5deb3",text="ACC",font='times 14')
lframe2=LabelFrame(frame1,bg="#f5deb3",text="B",font='times 14')
lframe3=LabelFrame(frame1,bg="#f5deb3",text="MAR",font='times 14')
lframe4=LabelFrame(frame1,bg="#f5deb3",text="MDR",font='times 14')
lframe5=LabelFrame(frame1,bg="#f5deb3",text="IR",font='times 14')
lframe6=LabelFrame(frame1,bg="#f5deb3",text="NF",font='times 14')
lframe7=LabelFrame(frame1,bg="#f5deb3",text="Instruction",font='times 14')
lframe8=LabelFrame(frame1,bg="#f5deb3",text="Comments",font='times 14')
lframe9=LabelFrame(frame1,bg="#f5deb3",text="Error",font='times 14')
PC = Label(lframe, text="", font = 'times 20', bg="#f5deb3", fg="BLACK")
ACC = Label(lframe1, text="", font = 'times 20', bg="#f5deb3", fg="BLACK")
B = Label(lframe2, text="", font = 'times 20', bg="#f5deb3", fg="BLACK")
MAR = Label(lframe3, text="", font = 'times 20', bg="#f5deb3", fg="BLACK")
MDR = Label(lframe4, text="", font = 'times 20', bg="#f5deb3", fg="BLACK")
IR = Label(lframe5, text="", font = 'times 20', bg="#f5deb3", fg="BLACK")
NF = Label(lframe6, text="", font = 'times 20', bg="#f5deb3", fg="BLACK")
Instruction = Label(lframe7, text="", font = 'times 20', bg="#f5deb3", fg="BLACK")
Comments = Label(lframe8, text="", font = 'times 20', bg="#f5deb3", fg="BLACK")
entry = Text(frame1, height=100, width=200, font='times 20')
style = ttk.Style()
style.configure("Treeview",font="times 13",background="#7fffd4")
style.configure("Treeview.Heading", font='times 13')
tree1 = ttk.Treeview(frame2, columns=("1"))
tree1.column('#0', width=180)
tree1.column('#1', width=180, anchor=S)
tree1.heading('#0', text="Address")
tree1.heading('#1', text="Label")
# example of adding element in tree :
# tree1.insert('','end', text="105H",values="X")
tree2 = ttk.Treeview(frame2, columns=("1","2"))
tree2.column('#0', width=120)
tree2.column('#1', width=120, anchor=S)
tree2.column('#2', width=120, anchor=S)
tree2.heading('#0', text="Address")
```

```
tree2.heading('#1', text="Label")
tree2.heading('#2', text="Value")
#SETTING DEFAULT VALUES IN RAM
for i in range(0,256):
    x=str(hex(i))[2:].upper()
    tree2.insert('','end',text=x, values=("NULL", "NULL"))
tree2.bind("<ButtonRelease-1>", EditTree)
entry1= Text(frame2, height=10, width=15, font='times 14')
entry2= Text(frame2, height=10, width=15, font='times 14')
entry3= Text(frame2, height=10, width=15, font='times 14')
entry3.config(state=DISABLED)
butt1.bind("<Button-1>", SubmitCode)
butt2.bind("<Button-1>", SetButton)
butt3.bind("<Button-1>", InsRead)
butt4.bind("<Button-1>", StopButton)

frame1.place(x=845, y=15, relwidth=1, relheight=1, height=-30, width=-800)
frame2.place(x=400, y=15, relwidth=1, relheight=1, height=-30, width=-1000)
frame3.place(x=15, y=15, relwidth=1, relheight=1, height=-30, width=-1060)

label20.place(x=100, y=7, relwidth=1, relheight=1, height=-660, width=-250)
label21.place(x=20, y=660, relwidth=1, relheight=1, height=-670, width=-355)
entry1.place(x=143, y=660, relwidth=0.92, relheight=1, height=-670, width=-300) #label
entry2.place(x=210, y=660, relheight=1, relwidth=0.92, height=-670, width=-300) #Decimal Value
entry3.place(x=71, y=660, relheight=1, relwidth=0.92, height=-670, width=-300)  #address
butt2.place(x=280, y=660, relwidth=1, relheight=1, height=-670, width=-300)
butt3.place(x=360, y=640, relwidth=0.8, relheight=1, height=-670, width=-300)
butt4.place(x=50, y=640, relwidth=0.7, relheight=1, height=-670, width=-300)


lframe.place(x=50, y=50, relwidth=0.5, relheight=0.1, height=-5, width=-160)
lframe1.place(x=205, y=50, relwidth=0.5, relheight=0.1, height=-5, width=-160)
lframe2.place(x=360, y=50, relwidth=0.5, relheight=0.1, height=-5, width=-160)
lframe3.place(x=50, y=130, relwidth=0.5, relheight=0.1, height=-5, width=-160)
lframe4.place(x=205, y=130, relwidth=0.5, relheight=0.1, height=-5, width=-160)
lframe5.place(x=360, y=130, relwidth=0.5, relheight=0.1, height=-5, width=-160)
lframe6.place(x=50, y=230, relwidth=0.38, relheight=0.1, height=-4, width=-160)
lframe7.place(x=35, y=330, relwidth=1.1, relheight=0.17, height=-4, width=-170)
lframe8.place(x=35, y=480, relwidth=1.1, relheight=0.15, height=-4, width=-170)

PC.pack()
ACC.pack()
B.pack()
MAR.pack()
MDR.pack()
NF.pack()
Comments.pack()
Instruction.pack()
IR.pack()

textfield.place(x=3, y=3, relheight=1, relwidth=1, height=-70)
```

```
butt1.place(x=0, y=650 , relwidth=1, relheight=0.5, height=-300)
tree1.place(x=20, y=40, relwidth=1, relheight=1,width=-42, height=-500)
tree2.place(x=20, y=280, relwidth=1, relheight=1,width=-42, height=-335)

Comments.configure(text='GG')
root.mainloop()
```