course name: computer organization and architecture.

project title: project 2 (timing diagram)

group members:
HISSAH 219025504
NOURAH 219010636
LATIFAH 219029556

submitted to: dr.Raazia saher

section:65

Content table

1.Introduction

In this project we decide to implement the timing diagram of register instructions using Java language. This program will cover all instructions that related to register instruction and will show each steps to execute the instruction the user want to execute from the fetching steps to the end of the instruction.

2.Problem statement

You are requested to design and implement a program (using any programming language), the program input would be a sequence of, at least, four instructions' codes in hexadecimal, the instruction codes should be taken from Morris Mano Model ISA (Instruction Set Architecture). Accordingly, the program should draw the timing diagram for the sequence of instructions' codes. The timing diagram should be drawn step-by-step (showing for T0, then For T0 and T1, then for T0, T1 and T2 , Then T0 …)

3.Register instruction

- **Programe:**

1)initial values
2)input
3)output

- **INITIAL VALUES**

PC=0 , E=0, S=0 And I is always zero

Register instruction is one of the most common type of computer instruction. It doesn't need to access the memory to take operand all operand and microopration done using the different type of registers. Register has different type of instruction which we will implement using Java language. Register instruction has the following instruction format.

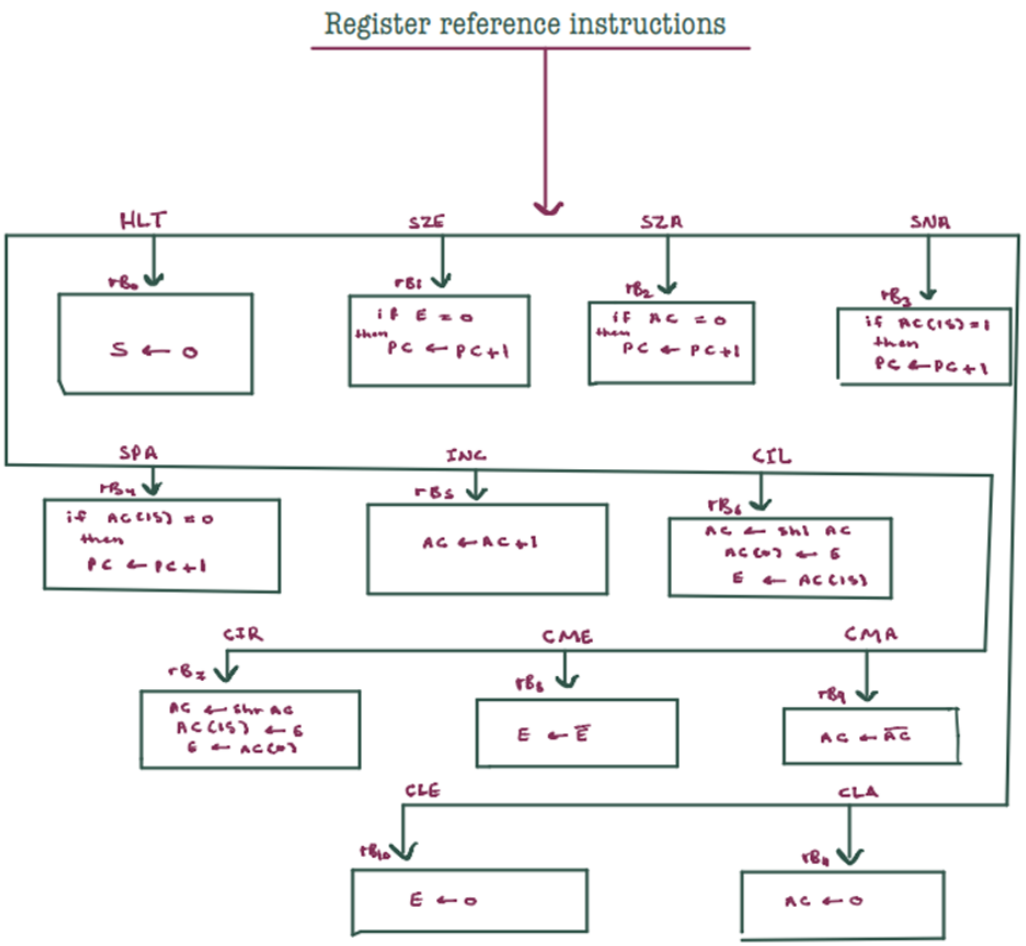
mode	opcode	microoperation

The mode bit will be 0 which equivalent to I’

And the opcode will be 111 which equivalent to D7 and in microoperation will have binary number represent the bit where the 1 will be there from B0 to B11. All instruction will complete their execution on T3.

Flowchart

The complete flowchart of the register instruction as a following:



- **INPUT**

1)The user choose either 1 or 2
1 to start with a program
2 to end with a program

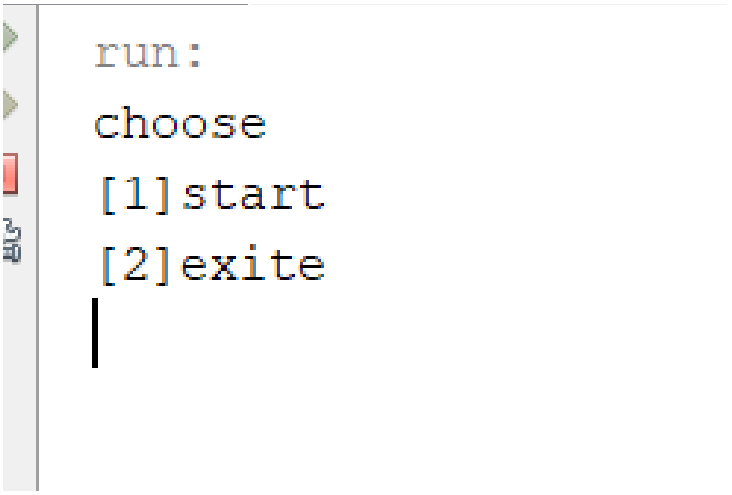
2)The user enter hexacode for register instruction such as 7800,7400,7200,7100,7080,7040,7020,7010,7008,7004,7002,7001

3)The user enters the initial value of AC

- **OUTPUT**

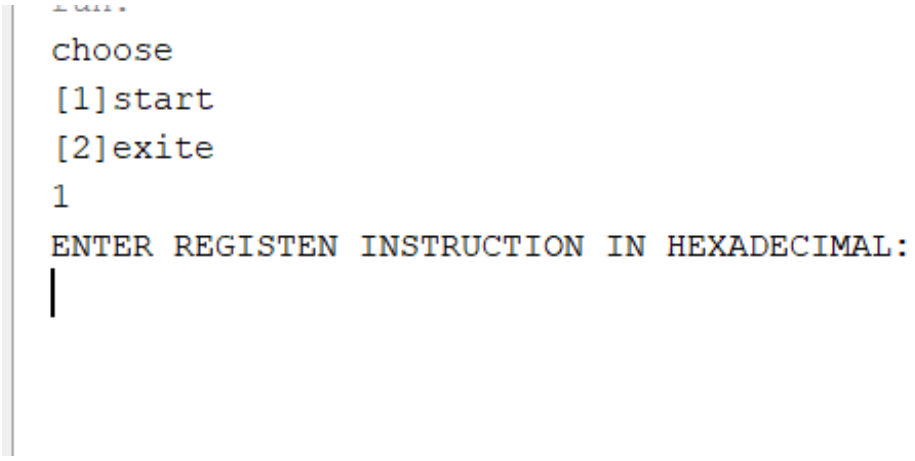
EXAMPLE +EXPLANATION

4.Running the program



The user choose either 1 or 2
1 to start with the program
2 to end with the program

When the user input 1
(start the program)



When the user enters the correct format

When the user click 1 the program asks for enter hexacode for register instruction such as
7800,7400,7200,7100,7080,7040,7020,7010,7008,7004,7002,7001

```
run:
choose
[1]start
[2]exite
1
ENTER REGISTEN INSTRUCTION IN HEXADECIMAL:
7800
this is register instruction
instruction is in binary: 0111100000000000
ENTER THE INTIAL VALUE OF AC IN DECIAML
```

After entering the hexacode the program converting hex code to binary code
e.g:
7800
I=0
7 for d7 =111
800=100000000000
Then the program asks to set an initial value for AC in decimal
It can be any number such as 270

When the user input incorrect format

```
run:
choose
[1]start
[2]exite
1
ENTER REGISTEN INSTRUCTION IN HEXADECIMAL:
38278`
incorrect entry :(
BUILD SUCCESSFUL (total time: 5 seconds)
```

When the user enters the correct format for ac

```
[1]start
[2]exite
1
ENTER REGISTEN INSTRUCTION IN HEXADECIMAL:
7800
this is register instruction
instruction is in binary: 0111100000000000
ENTER THE INTIAL VALUE OF AC IN DECIAML
270
THE ENTER VALUE IN BINARY EQUAVIELNT IS: 100001110
```

After entering 270 the program converted the 270 in decimal to binary code
100001110

When the user enters the correct format for ac(error accrue)

```
run:
choose
[1]start
[2]exite
1
ENTER REGISTEN INSTRUCTION IN HEXADECIMAL:
7800
this is register instruction
instruction is in binary: 0111100000000000
ENTER THE INTIAL VALUE OF AC IN DECIAML
klsk
error
BUILD SUCCESSFUL (total time: 5 seconds)
```

```
choose
[1]start
[2]exite
1
ENTER REGISTEN INSTRUCTION IN HEXADECIMAL:
7800
this is register instruction
instruction is in binary: 0111100000000000
ENTER THE INTIAL VALUE OF AC IN DECIAML
270
THE ENTER VALUE IN BINARY EQUAVIELNT IS: 100001110
THE CONTENT OF AC IS: 0000000100001110
```

100001110 less than 16 bit the program add more zeros to store it in AC because the ac store 16 bit

```
this is register instruction
instruction is in binary: 0111100000000000
ENTER THE INTIAL VALUE OF AC IN DECIAML
270
THE ENTER VALUE IN BINARY EQUAVIELNT IS: 100001110
THE CONTENT OF AC IS: 0000000100001110

SC<-0
```

After storing the initial value to ac, the program starts to fetch, decode and execute phases

Here we start with the first step of making SC =0 (start phase)

```
this is register instruction
instruction is in binary: 0111100000000000
ENTER THE INTIAL VALUE OF AC IN DECIAML
270
THE ENTER VALUE IN BINARY EQUAVIELNT IS: 100001110
THE CONTENT OF AC IS: 0000000100001110

SC<-0

T0 IS activate
the content of pc is: 0 after AR<-PC  the AR is: 0
```

(in fetching)

Second step t0, the program activates t0

AR<-PC

The initial value of pc =0 after doing this step to make AR<-PC

The content of AR became =0

```
this is register instruction
instruction is in binary: 0111100000000000
ENTER THE INTIAL VALUE OF AC IN DECIAML
270
THE ENTER VALUE IN BINARY EQUAVIELNT IS: 100001110
THE CONTENT OF AC IS: 0000000100001110

SC<-0

T0 IS activate
the content of pc is: 0 after AR<-PC  the AR is: 0

T1 IS activate
AFTER IR<-M[AR], PC<-PC+1 THE IR: 0111100000000000 THE PC: 1
```

(still in fetching)

After activating t0 now it is a time for t1, the program activates t1 which is

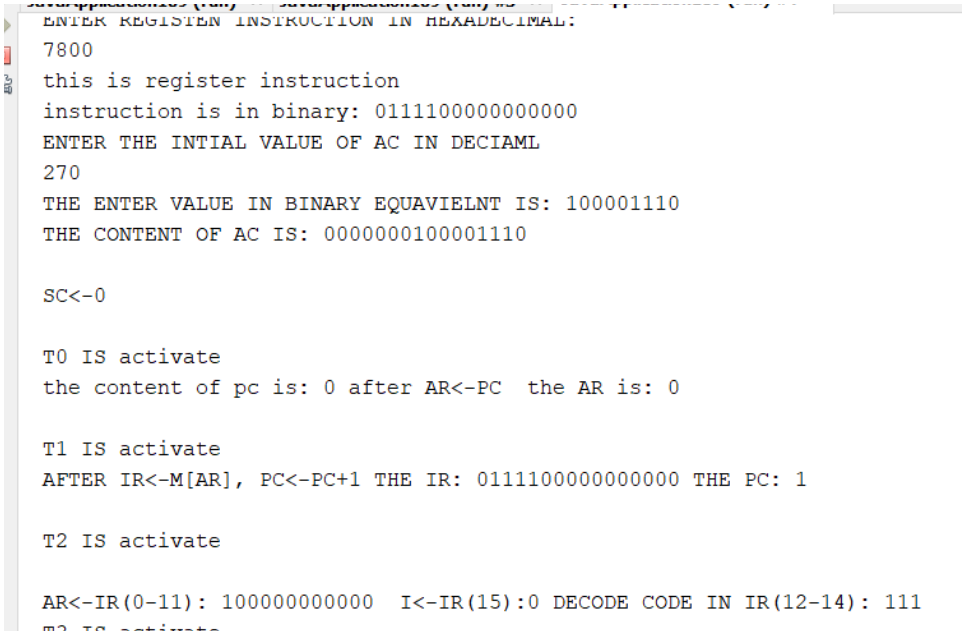
IR<-M[AR], PC<-PC+1

We know in the t0 phase the content of PC moves to AR and pc is incrementing.

AR holds the address of memory, now IR holds the instruction code

And pc is incremented to store the address of the next instruction

As we saw the previous value of pc was pc=0 after this step the pc became 1.



Here we finished with fetching now we start with decoding in t2

AR<-IR(0-11), I<-IR(15), DECODE CODE IN IR(12-14)

As we saw that IR is holding the instruction code here we store the address field to AR

AR = 100000000000

The I bit is always zero because we implement register instruction

I=0

and for decoding, we decode(12-14)bits for IR after decoding opcode (111)=d7.

d7=111

```
AR<-IR(0-11): 100000000000  I<-IR(15):0 DECODE CODE IN IR(12-14): 111
T3 IS activate
THE INSTRUCTION IS: AC<-0
after excute the instruction ........
after excute the instruction the content of ac is: 0000000000000000
after excute the instruction the content of e is: 0
after excute the instruction the content of s: 0

-----------------------------------------

               T0      T1      T2      T3
t0             1       0       0       0
t1             0       1       0       0
t2             0       0       1       0
t3d7I'         0       0       0       1
AC clr         0       0       0       1
AC inc         0       0       0       0
AC ld          0       0       0       0
pc inc         0       1       0       0

-----------------------------------------

choose
[1]start
[2]exite
```

After fetching and decoding, the type of instruction is specified which is AC<-0

Clear AC

Here in t3 is executing phase, the program shows the content of AC, E flipflop, and s flipflop

As we see after executing AC became 0000000000000000

And E=0 because the initial value of E=0 and because E is not affected in this instruction the value of E is not changed

And S=0 because the initial value of S=0 and because E is not affected in this instruction the value of S is not changed

```
-----------------------------------------

               T0      T1      T2      T3
t0             1       0       0       0
t1             0       1       0       0
t2             0       0       1       0
t3d7I'         0       0       0       1
AC clr         0       0       0       1
AC inc         0       0       0       0
AC ld          0       0       0       0
pc inc         0       1       0       0

-----------------------------------------
```

The output for the timing diagram
t0 =1
In AR<-PC
t2=1
In IR<-M[AR], PC<-PC+1
The pc inc in output became 1 because in this phase we increment pc
t3=1
our example is clear AC
So ac clr became 1

When the user input 2
(end the program)

```
choose
[1]start
[2]exite
2
thank you
BUILD SUCCESSFUL (total time: 2 minutes 4 seconds)
```

When the user inputs any number rather than 1 and 2

```
run:
choose
[1]start
[2]exite
732791
something went wronge :(
```

5.Conclusion

In our project, we have included all the register instructions, where the user can understand and deal with them easily by entering the instructions he wants. Also, the program shows him all the information related to the instructions and how to implement them and draw the timing diagram in a clear and simplified way.