Join the slack: <https://join.slack.com/t/architectureworkspace/shared_invite/MjM5MzYwNDcyMTYyLTE1MDUxNTMzMjItNjVhMTA1Y2E2Mw>

Commited Code to: <https://github.com/n1caps/Architecture>

Let me know your git names so i can add you as a contributor

Start of dev notes.

Started development of CISC Computer by coming up with a layout for the Front panel. Current iteration is more of a proof of concept/understanding so will not be doing anything too fancy. Just laying out all labels in a simple row and using radio buttons to denote bits being active (1) or not (0).

I have read the material the professor give. I think the most important problem is what GUI we use. I can understand the process of executing instruction but I don’t know how to achieve a panel. I have written some pseudocode, I don’t know if it helps.

Public class CPU{ //The main program, initiate all class  
 Private Register R0;// actually we can use class array, but I don't know whether there is class array in Java.  
 Private Register R1;  
 Private Register R2;  
 Private Register R3;// 4 GPR  
   
 Private Register X1;  
 Private Register X2;  
 Private Register X3;//3 Index Register  
  
 Private RAM;  
  
 //I don't know how to achieve addressing  
  
 Private Register PC;//Program Counter  
  
 Private Register CC;//Condition Code  
  
 Private Register IR;//Instruction Register  
  
 Private Register MAR;//Memory Address Register  
  
 Private Register MBR;//Memory Buffer Register  
  
 Private Register MSR;//Machine Status Register  
  
 Private Register MFR;//Machine Fault Register  
  
 Public class CPU(){  
 R0=new Register(16)  
 R1=new Register(16);  
 R2=new Register(16);  
 R3=new Register(16);  
  
 X1=new Register(16);  
 X2=new Register(16);  
 X3=new Register(16);  
  
 RAM=new Memory();  
  
 PC=new Register(12);  
 CC=new Register(4);  
 IR=new Register(16);  
 MAR=new Register(16);  
 MBR=new Register(16);  
 MSR=new Register(16);  
 MFR=new Register(4);  
   
 ER=new Erro\_Reporter();  
 DCD=new Decoder();  
 CT=new controller();  
 }  
   
 Public void main(){  
  
 }  
  
   
   
   
  
   
}  
  
Public class Erro\_Reporter {  
}  
Public class Decoder {  
}  
Public class Controller {  
}  
  
//!!All register cant call each other directly, call need to through Controller. Information is outputed in the Controller.  
Public class Register { //superclass of all register  
  
 Int Length;// bits of the register  
 String name=this.getName();//name of the class  
 Byte Memery[];//the memery of the register   
 int Pointer; //the pointer  
 int Flag;//the value of return   
  
 Public Register (int bits ){//initiate the class need the length of resgister  
 Length= bits;//initiate the length  
 Pointer= 0; //initiate the pointer  
 Memery=new byte[length];  
 }  
  
 Public Insert (byte data[], String source){ //fetch the data from source,data is array, source is the name of register  
 if (data.length==this.length){  
 for(i=0;i<this.lenght;i++){  
 this.Memery[i]=data[i];  
 }  
 Flag=0;//0 means complete  
 }  
 else if(data.length<data.length){  
 Flag=1;//1 means overflow  
 }  
 else if{data.length>data.length{  
 Flag=2;//2 means short  
 }  
 else {  
 Flag=3;//3 means some thing wrong  
 }  
 return Flag;  
 }  
  
 Public Output (){//the data is fetched from register  
 return this.Memery;  
 }  
   
}  
  
Public class Memory{  
 int length=2048; //2048\*16 bits  
 int word=16; //word is 16 bits  
 byte memory[];  
 int Pointer;  
 int Flage;  
   
 Public Memory(){  
 memory=new byte[word, length];  
 Pointer=0;  
 }  
   
 Public Insert (byte data[], String source){ //fetch the data from source,data is array, source is the name of register  
 if (data.length==this.length){  
 for(i=0;i<this.lenght;i++){  
 this.memery[i,this.Pointer]=data[i];  
 }  
 this.Pointer=this.Pointer+1  
 Flag=0;//0 means complete  
 }  
 else if(data.length<data.length){  
 Flag=1;//1 means overflow  
 }  
 else if{data.length>data.length{  
 Flag=2;//2 means short  
 }  
 else {  
 Flag=3;//3 means some thing wrong  
 }  
 return Flag;  
 }  
  
 Public Output (){//the data is fetched from register  
 return this.memery;  
 this.Pointer=this.Pointer-1;  
 }  
}

In my design, every register is instance of class Register. Memory is a special register, because it is two-dimension array(16bit\*length). Every register have two methods Insert and output, and register only can be called by class Controller. Another important class is Decoder. Decoder decodes the instruction and tell the Controller how to do, then Controller call the Registers. We maybe need to write one method in the Controller for each instruction.



Omoniyi 9/11/17 - Hopefully Tagging things like this will make it easier to track  
To operate the computer as of this posted date do the following:

1. Press the IPL button this will initialize the PC, and update the MAR, MBR, and IR registers with values based on the PC.
2. Computer will attempt to start executing the instruction in the IR (right now it is halt so it will simple halt)

To test your Instruction (Should be done after pressing the IPL button):

1. Pressing will the “Single Step” button will execute whatever instruction that is at the location pointed to by the PC.
2. So simply enter the instruction you wish to Execute into the switch register and hit “Deposit” to get it into memory at the address in the MAR register (this should be the same location as the PC).
3. Once the the instruction is in memory you can the press the “Single Step” button to execute the instruction (and increment the PC by 1)

To reset the PC just toggle the IPL button off and then back on

Omoniyi 10/2/17-Cache thoughts

I believe that cache is essentially the same thing as main memory but much smaller

I think we should have the it as a separate class called cache

The class should create a cache object which is essentially an array of 16 lines, each line should contain the address in memory that it is pointing and the data found at that location

PsuedoCode for cache object:  
ArrayList<int[]> cache = new ArrayList<int>()

Whenever you get are trying to retrieve something from memory you first check to see if that memory location is in the cache:  If it exists pull the data from the cache; if it does not exist in the cache go to main memory, attempt to get the data from main memory and place it in the data and location at the end of the cache arraylist; if the arraylist is full (cache.size() > 16) then remove the first item in the list.

Kevin Zhu [2:21 AM]

added this Plain Text snippet: Descript.txt

BD:best difference T1:Temp 1 T2:Temp 2 ND:new differece

i:counter ONE: increment MI:maximum iteration

BV:best value NB:normal bias AB:array bias

X1<-c(26) //[LDX x=0 i=0 address=26]Load Value of NB;!depend on position of NB

X2<-c(27) //[LDX x=0 i=0 address=27]Load Value of AB;!depend on position of AB

R0<-c(29) //[LDR r=0 x=0 i=0 address=29]Load Value of INPUT;!depend on position of INPUT

R0<-R0-c(c(28)+X2) //[SMR r=0 x=2 i=1 address=28] Caculate Euclidean distance from INPUT to N(i);!the address is depended on position of i

If(R0)>=0 jump to 14;else pc++ //[JGE r=0 x=0 i=0 address=14] if >=0 means R0>c(N(i)) so jump to line 14,if <0 means R0<c(N(i)),need to exchange their position and do it again; ! the jump position depend on the position of next section

M(30)<-R0 //[STR r=0 x=0 i=0 address=30]Give the value to T1;!the address depend on the position of T1

R0<-c(c(28)+X2) //[LDR r=0 x=2 i=1 address=28]Give the array value to R0;!the address depend on the position of i

R0<-R0-c(30) //[SMR r=0 x=0 i=0 address=30]Now R0 is new difference;!the address depend on the position of T1

R0<-R0-c(32) //[SMR r=0 x=0 i=0 address=32]Caculate the difference between new difference and old difference;!!the address depend on the position of BD

If(R0)>=0 jump to 20;else pc++//[JGE r=0 x=0 i=0 address=20] if >=0 jump,else exchange value and do it again;!the address depend on the position of R0<-c(i)next section

R0<-c(X1+13) //[LDR r=0 x=1 i=0 address=13]ND at 33;!depend on position of ND

M(X1+12)<-R0 //[STR r=0 x=1 i=0 address=12]Update best difference,BD at 32;!depend on position of BD

R0<-c(c(28)+X2) //[LDR r=0 x=2 i=1 address=28]i at 28;!depend on position of i

M(X1+15)<-R0 //[STR r=0 x=1 i=0 address=15]BV at 35 Update best value;!depend on position of BV

R0<-c(28) //[LDR r=0 x=0 i=0 address=28]i at 28;!depend on position of i

R0<-R0+c(X1+14) //[AMR r=0 x=1 i=0 address=14]ONE at 34,i++;!denpend on position of ONE

M(i)<-R0 //[STR r=0 x=0 i=0 address=28]Update i;!denpend on position of i

R0<-R0-c(X1+16) //[SMR r=0 x=1 i=0 address=16]MI at 36,Judge whether break iteration;!denpend on position of MI

if c(R0)!=0 jump 8; else pc++ //[JGE r=0 x=0 i=0 address=8]if not jump to line 8

Jump to X1+27 //[JCC cc=1 x=1 I=0 address=27] jump across valuables;!depend on position of R0<-c(BV)

NB=20 //the bias

AB=37 //the bias of array;!depend position of array

i=1

INPUT

T1

T2

BD

ND

ONE

BV

MI

N1

N2

N3

N4

N5

N6

N7

N8

N9

N10

N11

N12

N13

N14

N15

N16

N17

N18

N19

N20

R0<-c(BV) //[LDR r=0 x=1 i=0 address=15]BV at 35,Give the best value to R0;!denpend on the position of BV

OUT(R0) //[OUT r=0 devid=1]Output the final value

11/1/17

Kevin Program 2 psuedocode:

1.First,we need a functionality to load the paragraph from file to Memory.(Start position at the Memroy depend on the register PC)

2.We need to create a ASCII table( or we can use Java to convert)

3.Every memory cell stores one letter.

Constants:

The number of words of Paragraphs: Psize

The number of words of Input: Isize

The start position of Paragraphs array: Pbias

The start position of Input array: Ibias

Variable:

The number of Paragraphs: Pnum #each "\n" Pnum++

The number of Words: Wnum #each " " Wnum++

The counter of Input: j #when j==Isize means it has found the word in paragraph

The counter of Paragraphs: i

Array:

The Input: Input[]

The Paragraphs: Paragraphs[]

j=0;

Pnum=0;

Wnum=0;

for(int i=0;i<Psize;i++){

if(j==Isize){

return [Pnum,Wnum];

}

if(Input[j]==Paragraph[i]){

j++;

}

else{

if(Paragraph[i]=="\n"){

Pnum++;

}

else if(Paragraph[i]==" "){

Wnum++;

}

else{

j=0;

}

}

}

[7:22 AM]

We may need to use interface Runnable and Class ThreadLocal. We need to separate instruction identification and instruction execution. Whenever instruction is branch instruction, creating a new thread and copying all variables by ThreadLocal. If prediction is correct and same, killing the original thread. If prediction is wrong, killing the prediction thread.