Lab 8 report Computer Architecture

- CS21BTECH11011
- B. Asli Nitej Reddy

Basic outline:
Input is in a file named “lines.txt” which contains machine code line by line

I am counting total no of lines in that file for creating arrays of type string and int and after completing counting I am closing it again

After completion of executing (string array contains the equivalent dissembles code and int arrays contains the same values as line if any label starts their else value is zero)

Now again I opened file for reading the file “lines.txt” line by line and at start of while loop each line is copied into a char array of size 8

Now I am sending this array into a function named hex_to_bin (takes input as char [] and returns a string (contains the equivalent binary code of that hexadecimal code which I sent using char array)) which returns a string, and I am storing that value of string into a string named b

Now we have the equivalent binary code of that hexadecimal code in a string named b

Using the last 7 digits (indirectly opcode) we are creating 8 cases which goes as follows
1.R type of (add, sub , and , or , xor , sll , srl , sra)
2.I type of (addi , andi , ori , xori , slli , srli , srai)
3.I type of (ld , lw , lh , lb , lwu , lhu , lbu)
4.S type of (sd , sw , sh , sb)
5.B type of (beq , bne , blt , bge , bltu , bgeu)
6.J type of (jal)
7.I type of (jalr)
8.U type of (lui)

R format instructions:
For R type using func3 (17 to 19) and func7 (0 to 6) i.e., again reading the bits from string b and again making different cases similarly finding rs1, rs2, rd from different locations and using all these I finally created a string called final which contains equivalent assembly code and storing that final into string arrays at location i where i = line number

I format instructions:
For I type there are 3 possible opcodes based on last 7 bits
Case1: 0010011 which contains (addi , andi , ori , xori , slli , srli , srai)
Case2: 0000011 which contains (ld , lw , lh , lb , lwu , lhu , lbu)
Case3: 1100111 which contains (jalr)

Case1:
Here based on func3(17 to 19) again making different cases and immediate for addi , andi , ori , xori is first 12 bits from string b and with sign included whereas for others in this case(slli , srli , srai) immediate value is from(6 to 11) without sign included and based on immediate (0 to 6) and func3 slli , srli , srai are differentiated
Case2:
Here we will have all load types based on func3 we will differentiate between them func3 is bits from (17 to 19) location in string b
Case3:
Only jalr is present

In all above cases I am storing the final calculated assembles string into string array at location i where i is the line number

S format instruction:
For S format using func3 (determined he cases) (17 to 19) location in string b and calculated rs1, rs2 and immediate value (concatenated two strings string1(0 to 6) and string2(20 to 24) from string b)

Here also I am storing the final calculated assembles string into string array at location i where i is the line number

For others formats :
Similar types of approach is used (like above three for immediate different concatenation and for any extra inner cases (use of func3 or func7 if exists) and also same for calculating register numbers using different bits from string b) for the B format and J format and U format instructions

How I am kept tracking of labels which will appear in B and J format:
For B format and J format after calculating immediate (using concatenation and other techniques) I am going to the int array let that array named be branches the I am going to location branches [i + (k\4)] and making that value from (0) to (i + (k\4)) to keep tract of labels

Final printing of assembled code:
After going to every line, I will have two arrays one is of integers which contains the value i where i is the line number if any label starts there else the value is 0 and the other is of strings which contains the assembled code

And the label name will be like L'I' where i is the line number if any lable starts there and I am printing using for loop

What I did to test code:
I copied a code from one of my exercises into into the website (https://venus.kvakil.me) and changed the doubles to single words as that website not accepting 64 bit format and then I dumped the machine code generated from there into the input file named "lines.txt" then I ran my program and compared output produced in the terminal with respect to ripes dissembled code other than label names every thing came same as label names we can keep anything like this I tested my code