Col 216

Assignment 3

2018CS10351

2018CS10371

The approach :

We have created a vector named instructions that reads the assembly language instructions and stores in it in order.

After all the instructions have been read we initialize a array with size equal to 218 - size of the vector instructions as the memory size is 220 bytes and we are storing 4 bytes in 1 single array element.

We have a program counter(PC) which is initialized to 0 and depending on the instructions read its value is determined at the end of every instruction.

We have assumed all the registers to be integral and no register as special or something of that sort.

There are 32 registers which are named same as in mips like ($zero,$at,$v0,$v1,$a0,$a1,$a2,$a3,$t0,$t1,$t2,$t3,$t4,$t5,$t6,$t7,$s0,$s1,$s2,$s3,$s4,$s5,$s6,$s7,$t8,$t9,$k0,$k1,$gp,$sp,$fp,$ra).

we have stored every instruction in the form of a struct which has to attributes the operation and the operands.

Format for various instructions :

add : dest\_r1,source\_r2,source\_r3

sub : dest\_r1,source\_r2,source\_r3

mul : dest\_r1,source\_r2,source\_r3

beq : source\_r2,source\_r3,line\_number\_to\_jump

bne : source\_r2,source\_r3,line\_number\_to\_jump

slt : dest\_r1,source\_r2,source\_r3

lw : register,address

sw : register,address

j : line\_number\_to\_jump

addi : source\_r2,source\_r3,value\_to\_add

Error conditions :

when we are execution every instruction then based on the operator that is encountered we take all the attributes and if the operand does not look similar then the simulator returns syntax error.

Also we have handled the cases where if the number of operands for a given operator is less or more than what is expected then we return a error .

Also if the register that is given is not recognized by the simulator then we give an error .

If in case of a jump instruction if the line number is not a integer then we return a error .

Like wise if the second operand in lw and sw is not integer and a multiple of 4 then we return a error .

If the third operand in addi is not integer then we return an error.

The instructions in which jump is involved give an error if the instruction they ask for jump is out of bounds to the memory.