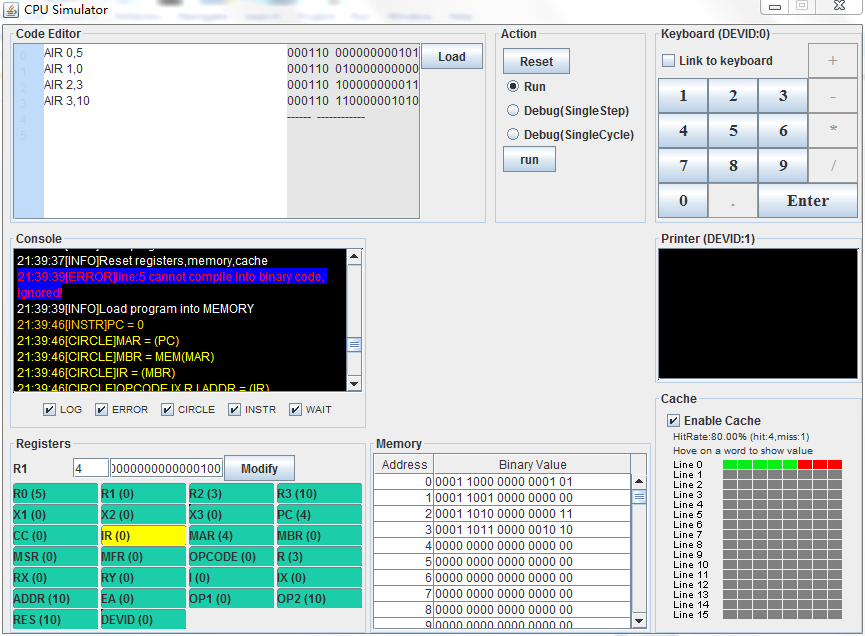
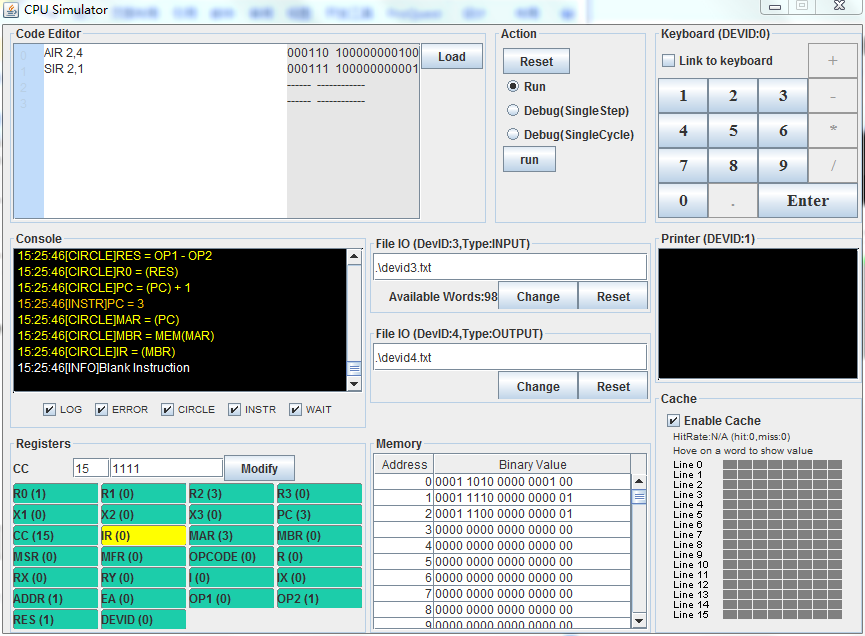
**006 AIR r immed: Add immediate to Register**



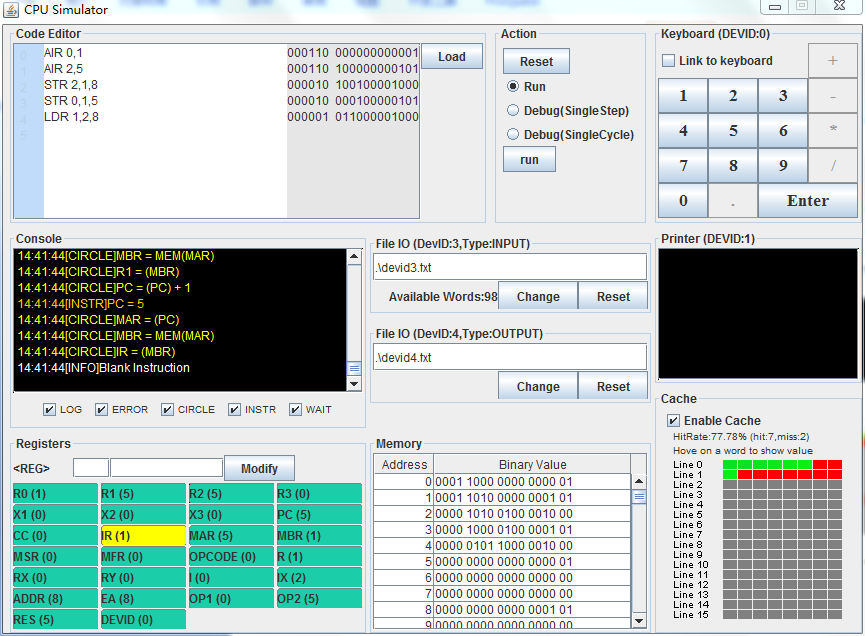
If immed = 0, the Console shows ERROR, and the immed cannot load in register (the R1 contains nothing)

**007 SIR r, immed: Subtract Immediate from Register**

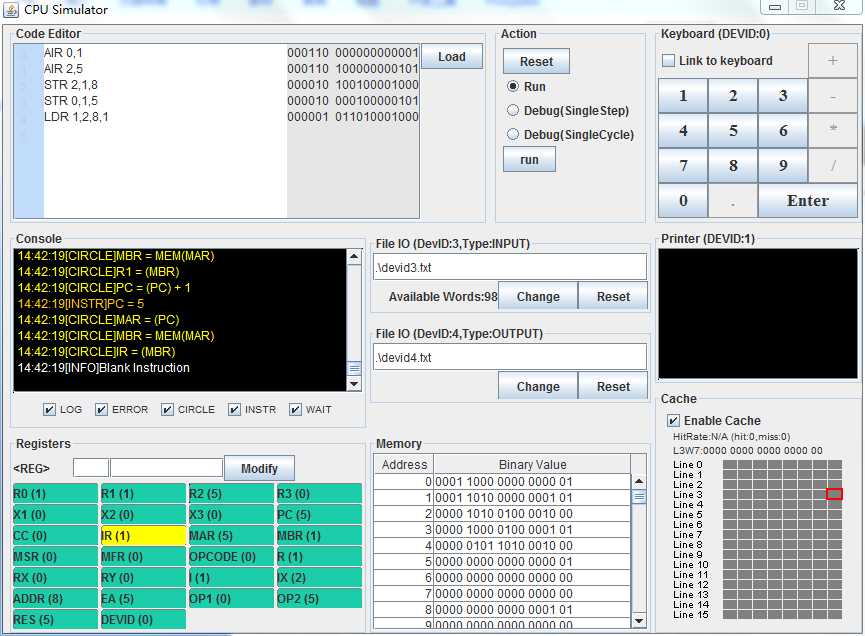


R<-c(r) - Immed

**01 LDR r, x, address[,I]: Load register from memory.**



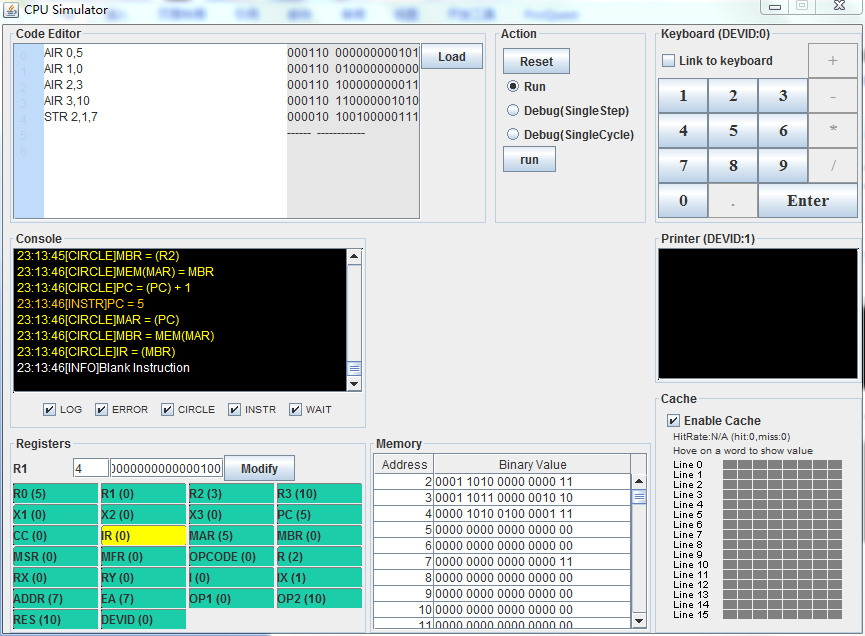
Put the content (value 5) of the memory whose address is 8 into the register (R1). The offset in X2 is 0, so EA is 8. Therefore, the value 5 was put into the R1.



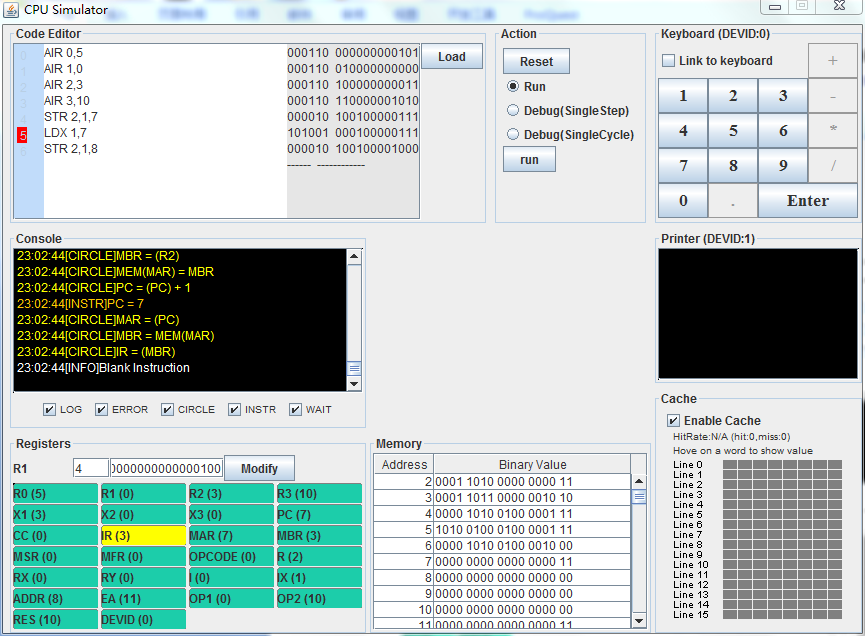
If we set I bit = 1, the value 5 would be the address and the value 1 was put into the R1.

**02 STR r, x, address[,I]: Store register to memory**

**41 LDX x, address[,I]: load index register from memory**

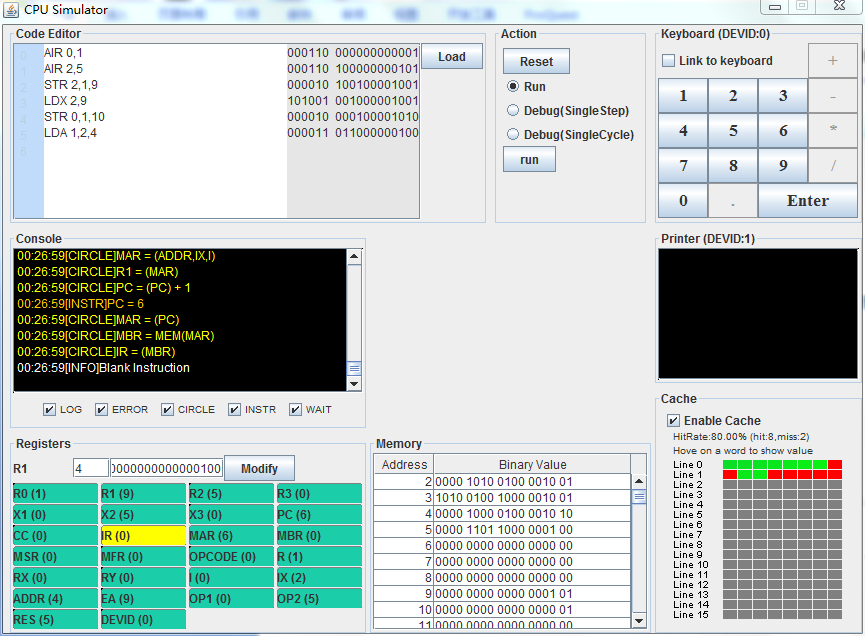


Put the content (value 3) in R2 into the memory whose address is 7 with the offset is 0 (in X1).



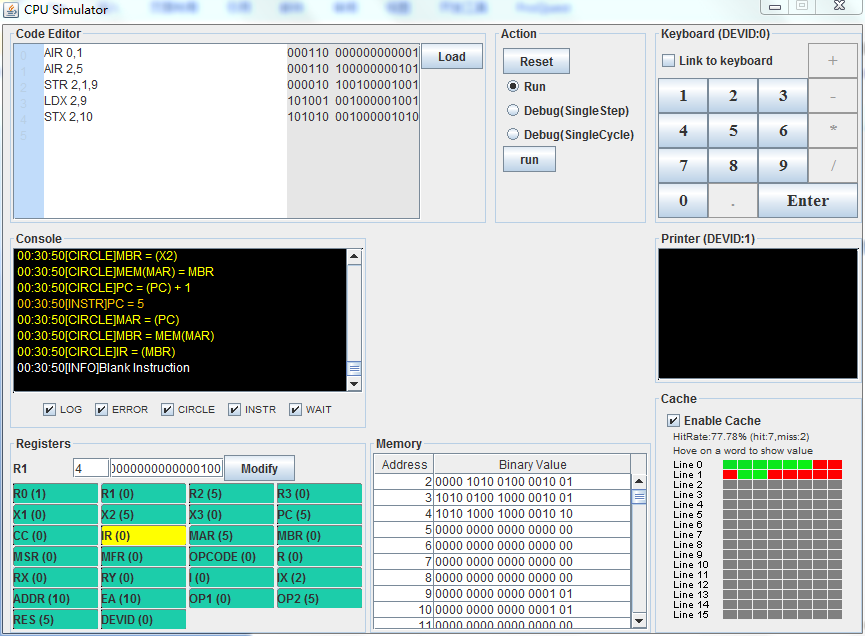
Load index register (X1) from memory whose address is 7, so the X1 was loaded with 3. Put the content in R2 into the memory whose address is 8 with the offset is 3 (in X1), but the effect address (EA) is 11. So the memory whose address is 11 contains value 3.

**03 LDA r, x, address[,I]: Load register with address**



Put the content (value 5) of the memory whose address is 4 into the register (R1), but the offset in X2 is 5, so EA is 9. Therefore, the address 9 was put into the R1.

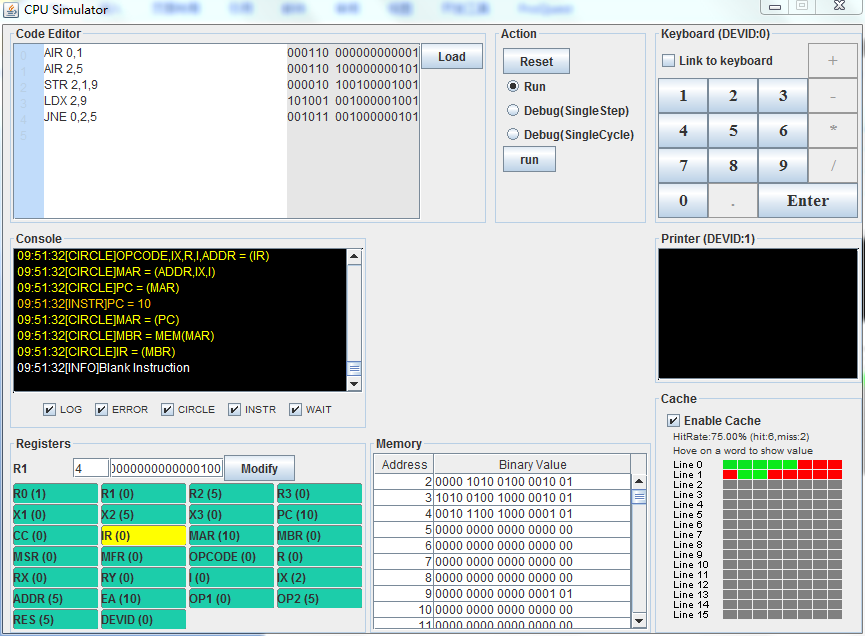
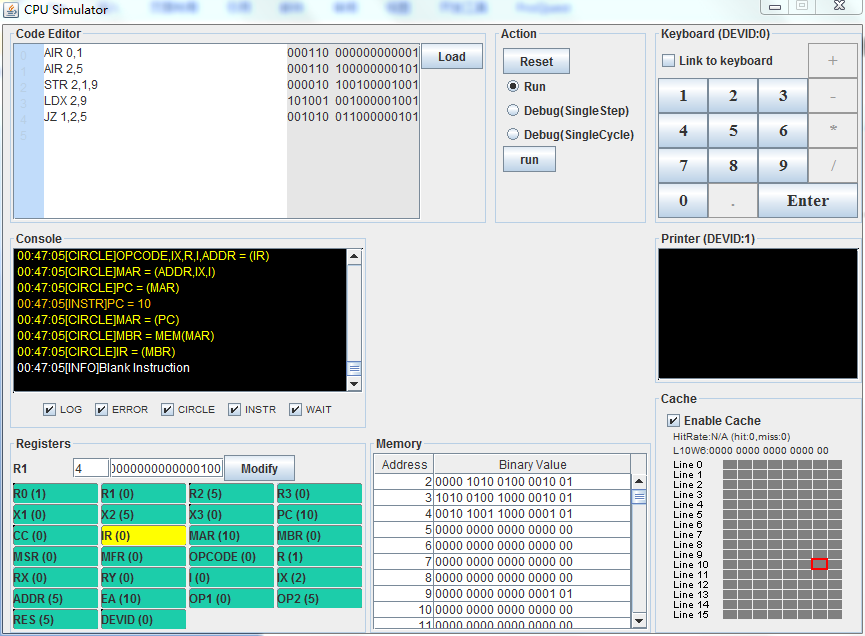
**42 STX x, address[,I]: store index register to memory**



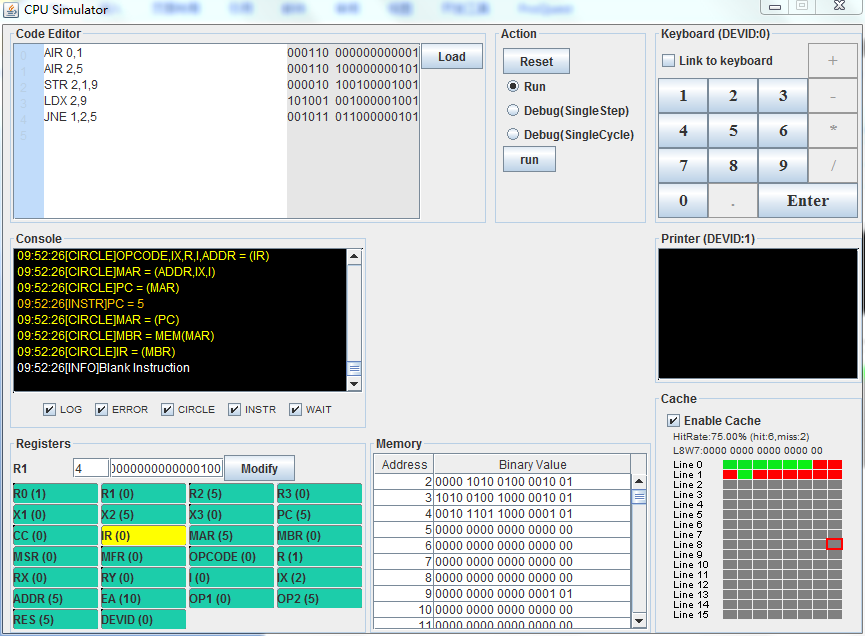
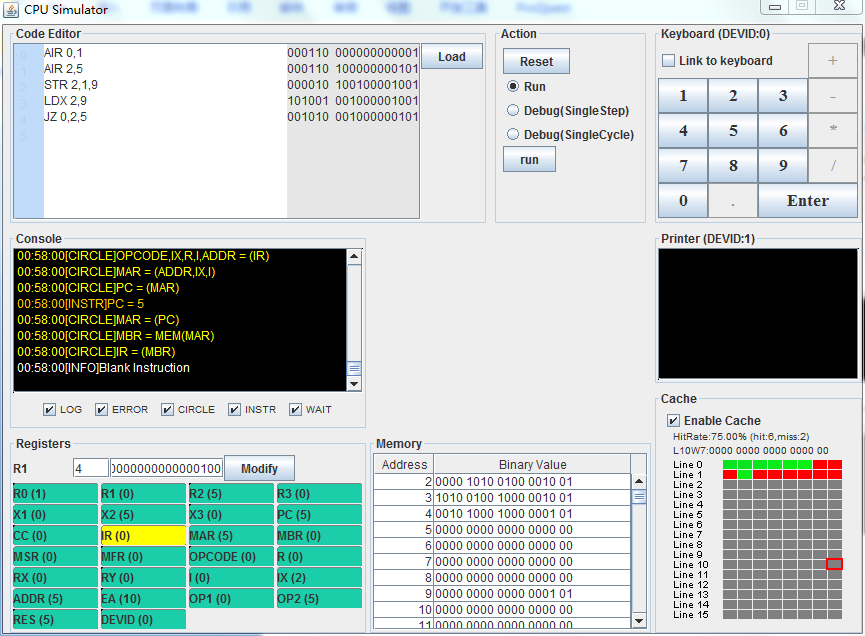
Put the content (value 5) in X2 into the memory whose address is 10, so the content in memory whose address is 10 is value 5.

**010 JZ r, x, address[,I]: jump if zero**

**011 JNE r, x, address[,I]:jump if not equal**

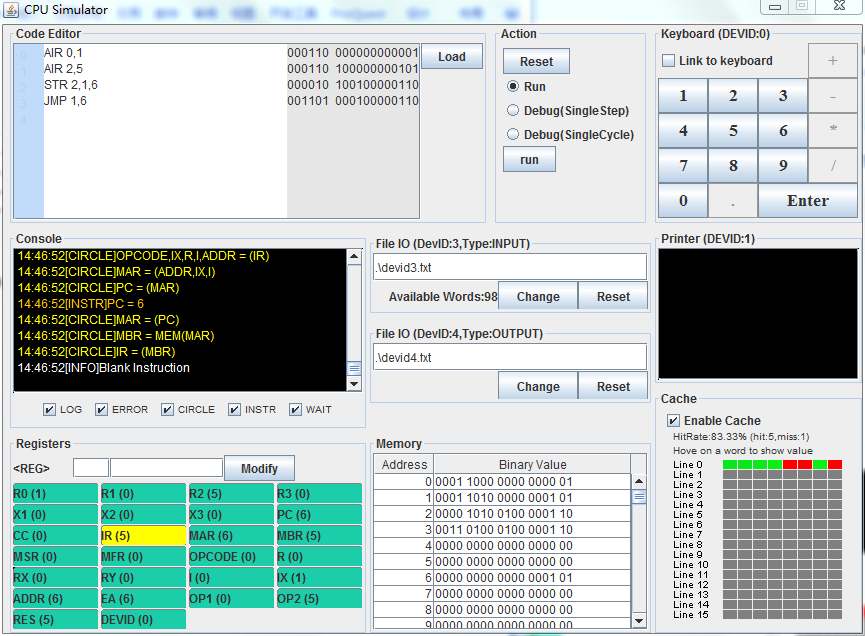


For JZ, when c(R) = 0 (R1 = 0), the EA (value 10 = 5 + 5) was put into PC. Contrast to JZ, for JNE when c(R) != 0 (R0 = 1), the EA (value 10 = 5 + 5) was put into PC.

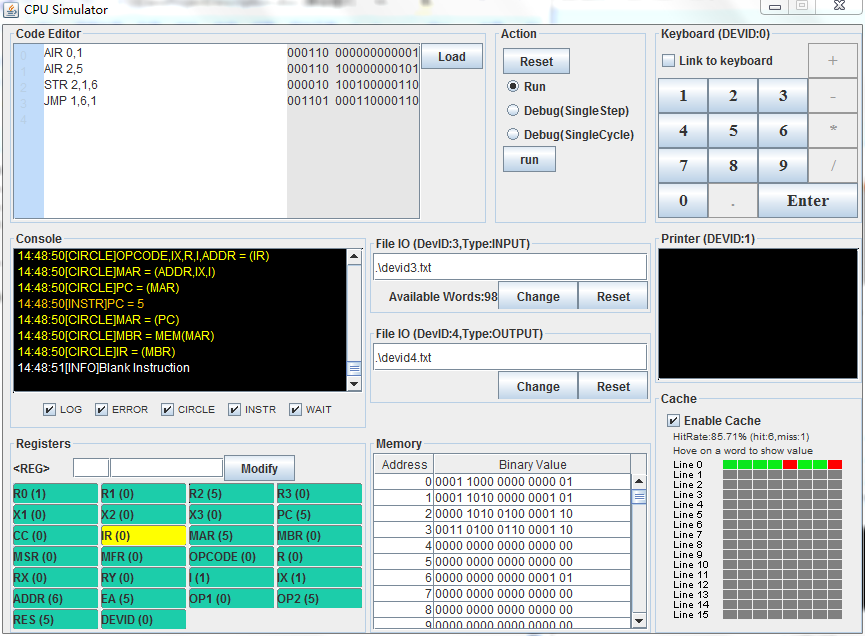


For JZ, when c(R) != 0 (R0 = 1), PC = PC + 1 (value 5 = 4 + 1). Contrast to JZ, for JNE when c(R) = 0 (R1 = 0), PC = PC + 1 (value 5 = 4 + 1).

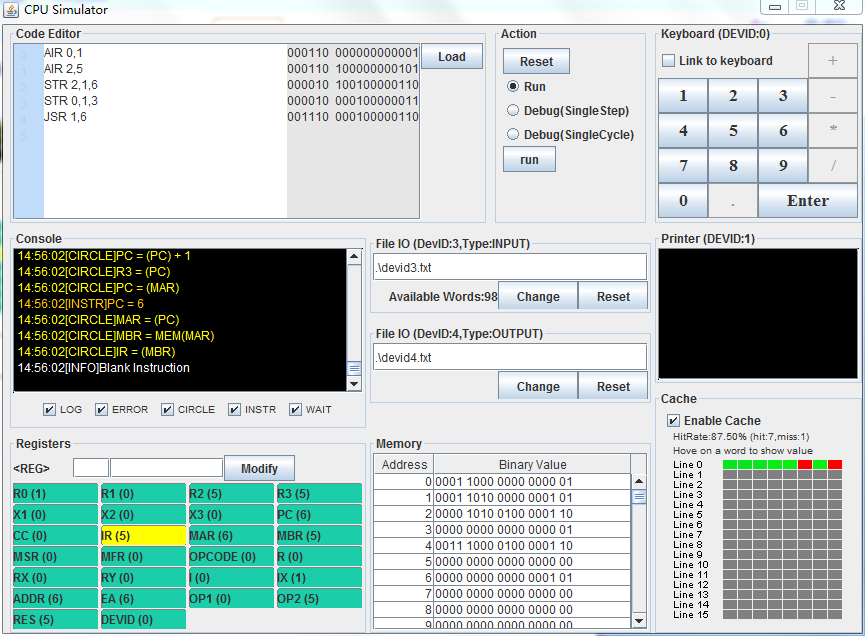
**013 JMP x, address[,I]:** **Unconditional Jump To Address**



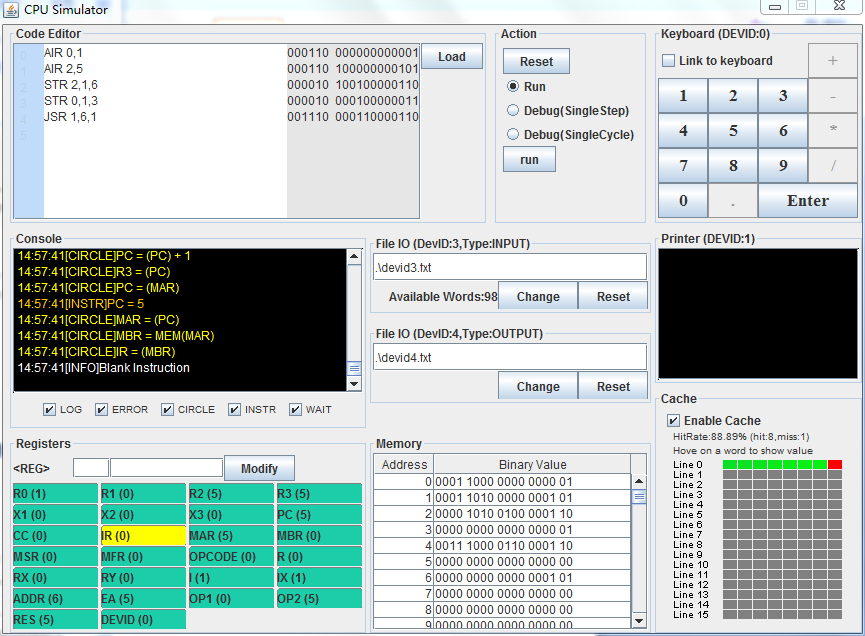
PC <- EA (value 6), if I bit not set;

PC<- c(EA), if I bit set

**014 JSR x, address[,I]: Jump and Save Return Address**

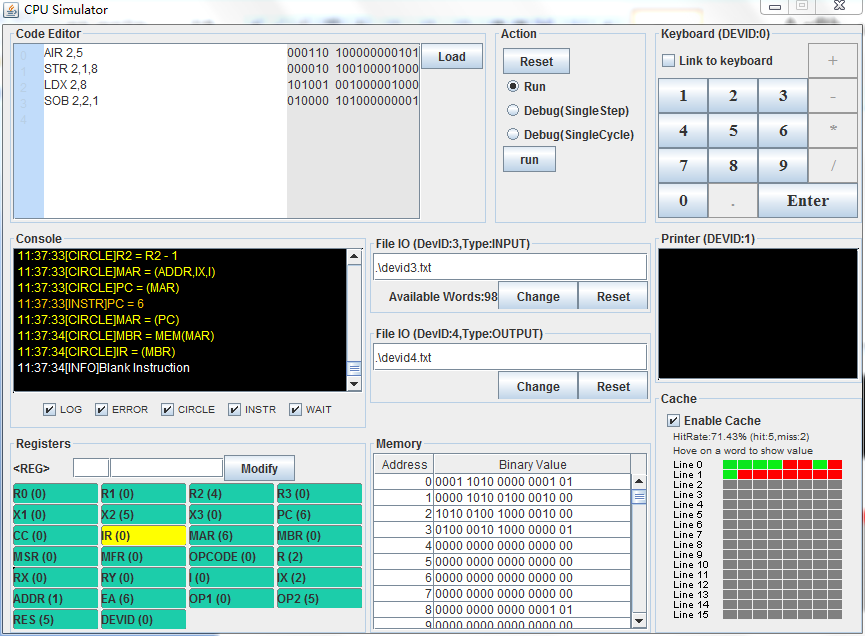


R3<-PC+1; PC<-EA

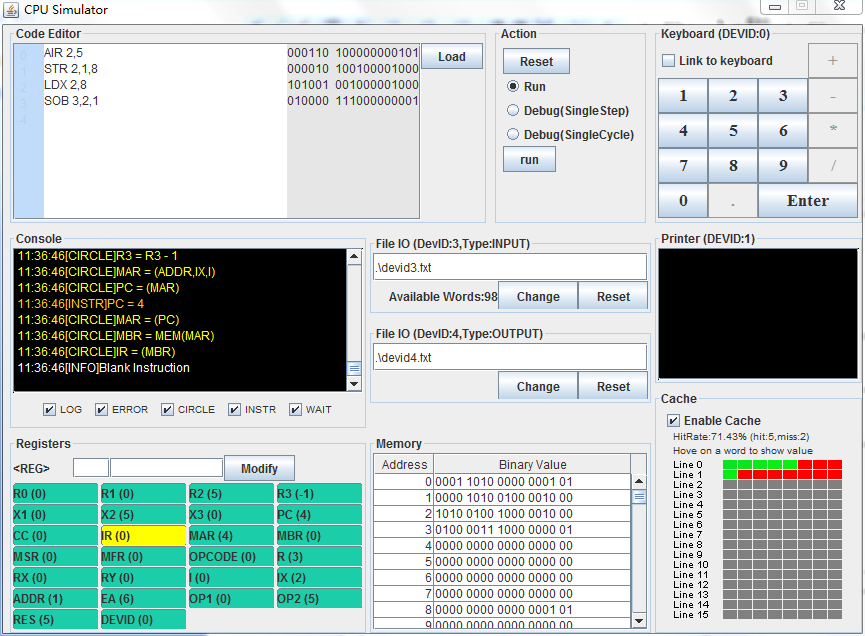


PC <- c(EA), if I bit set.

**016 SOB r, x, address[,I]: Subtract One and Branch.**

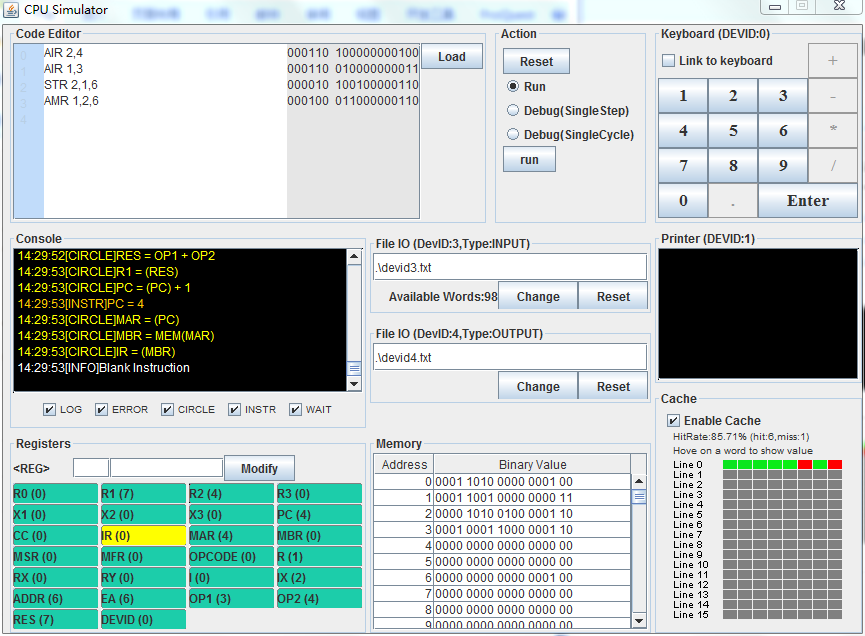


The value of R2 subtracts one and then returns the result to R2 (value 4 = 5 - 1). The value in R2 > 0 (value 4), so put the EA (value 6) into PC.



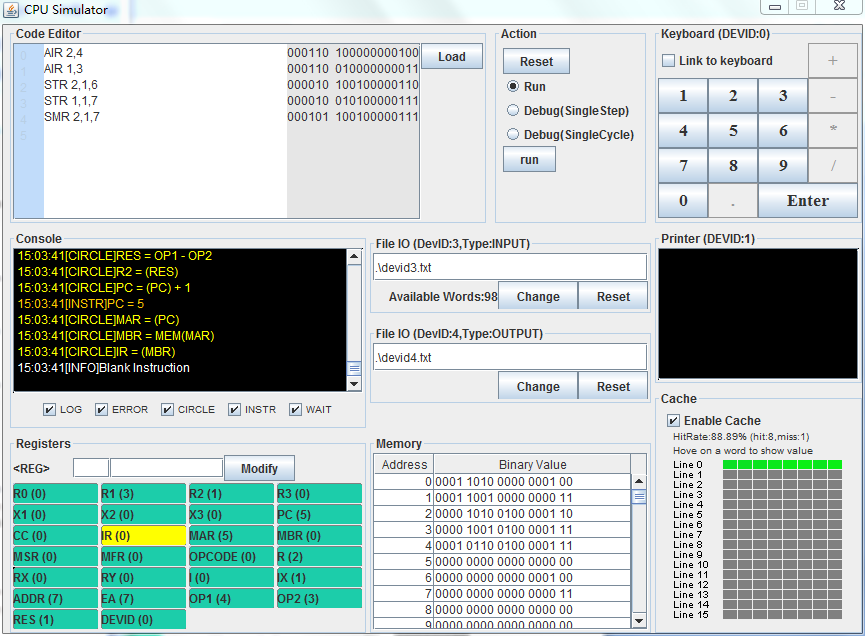
The value of R2 subtracts one and then returns the result to R3 (value -1 = 0 - 1). The value in R3 < 0 (value -1), so PC = PC +1 (value 4).

**004 AMR r, x, address[,I]: Add Memory To Register**



The value (value 4) of memory was added to the R1 (value 3), so the R1 store value 7 (7 = 4 + 3).

**005 SMR r, x, address[,I]: Subtract Memory From Register**



R<-c(r) – c(EA).

**061 IN r, devid: Input Character To Register from Device**

**062 OUT r, devid: Output Character to Device from Register.**

