```
       ORG 0000
       MOVE NUMBER,B *reset random num to zero*
back   CMP #$0,B
       BNE BACK
       MOVE B,NUMBER
       JSR Random
Again  JSR firstdigit
       JSR SecondDigit
       MOVE NUMBER,B
       MOVE 1stDigit,A
       MOVE 2ndDigit,A
       CMP A,B
       BEQ CORRECT
       BNE go
go     JSR COMPARE
       JMP BACK2 *COMPARE THEN LOOP AGAIN TILL EQUAL THEN DISPLAY CORRECT
       HALT

BACK SUB #$1,B
     JMP back

BACK2 MOVE B,NUMBER
      JMP Again
**************************************
Random MOVE $00E8,B *SET RANDOM NUM VAR*
       AND #^16,B
       MOVE B,NUMBER *MOVE TO VARIABLE*
       RTS        *return to main*

**************************************

firstdigit MOVE $E1,A *CHECK IF KEYBOARD READY
           CMP #$0,A  *check to see if its ready
           BEQ firstdigit *go back if not ready
           MOVE $E0,A *READ FROM KEYBOARD
           CMP #^49,A *check for a 1
           BNE checkZero

           JSR addingOne  *if 1 make it 10 for first digit
           JMP FirstEnd *display

checkZero CMP #^48,A *checks for a 0 *not equal to 0 go back *
          BNE firstdigit
          JMP FirstEnd *display zero

FirstEnd  MOVE $E3,B *check if console output is ready
          CMP #$0,B
          BEQ FirstEnd
          MOVE A,$E2 *WRITE TO CONSOLE
          RTS
*********************************************
SecondDigit
wait2  MOVE $E1,A *CHECK IF KEYBOARD READY
       CMP #$0,A  *check to see if its ready
       BEQ wait2 *go back if not ready
       MOVE $E0,A *READ FROM KEYBOARD
       CMP #^49,A *check for a 1
       BNE check1
       JSR adding
       JMP wait3

check1 CMP #^48,A *checks for a 0 *not equal to 0 go back *ALLOW USER TO INPUT 0-9
       BNE check2
       JMP wait3

check2 CMP #^50,A *2
       BNE check3
       JSR adding1
       JMP wait3

check3 CMP #^51,A *3
       BNE check4
       JSR adding2
       JMP wait3

check4 CMP #^52,A *4
       BNE check5
       JSR adding3
       JMP wait3

check5 CMP #^53,A *5
       BNE check6
       JSR adding4
       JMP wait3

check6 CMP #^54,A *6
       BNE check7
```

```
       JSR adding5
       JMP wait3

check7 CMP #^55,A *7
       BNE check8
       JSR adding6
       JMP wait3

check8 CMP #^56,A *8
       BNE check9
       JSR adding7
       JMP wait3

check9 CMP #^57,A *9
       BNE wait2
       JSR adding8
       JMP wait3

wait3  MOVE $E3,B *check if console output is ready
       CMP #$0,B
       BEQ wait3
       MOVE A,$E2 *WRITE TO CONSOLE
       RTS
*******************************************************
*COMPARING USER INPUT AND RANDOM VAR
*******************************************************
COMPARE
       MOVE A,INPUTS *CLEARING A
compar CMP #$0,A
       BNE taking1
       MOVE 1stDigit,A
       MOVE B,BINPUTS *clears B
       MOVE 2ndDigit,B
       ADD B,A
       MOVE B,BINPUTS*CLEAR
compar1 CMP #$0,B
        BNE takingB
        JSR workout *WORKOUT IF A IS LOWER,HIGHER OR CORRECT TO B / RANDOM NUMBER
        RTS
*A AND B TO BE EQUAL TO ZERO  LOOP*

takingB SUB #$1,B *TAKE 1 TILL EQUAL TO 0 B
        JMP compar1


taking1 SUB #$1,A *TAKE 1 TILL EQUAL TO 0 A
        JMP compar

**********************
*DISPLAY4COMPAR*
**********************

workout MOVE NUMBER,B
        CMP A,B
        BEQ CORRECT
        CMP A,B
        BMI LOWER *take A away from B if A is bigger than B then minus left so display Lower
        CMP A,B
        BMI HIGHER
        RTS

CORRECT MOVE A,DUMP
        MOVE B,DUMP
correct MOVE $E3,B *check if console output is ready
        CMP #$0,B
        BEQ correct
        MOVE #$67,A  *Move ASCII C TO A to be displayed CORRECT
        MOVE A,$E2   *WRITE TO CONSOLE/DISPLAYS
        JSR SATTEMPTS *shows attempts

HIGHER  MOVE B,NUMBER* put variable value back
        MOVE A,DUMP
higher  MOVE $E3,B *check if console output is ready
        CMP #$0,B
        BEQ higher
        MOVE #$72,A  *Move ASCII H TO A to be displayed HIGHER
        MOVE A,$E2   *WRITE TO CONSOLE/DISPLAYS
checki2 CMP #$0,A
        BNE subby2
        ADD #$1,A
        MOVE A,ATTEMPTS  *every higher or lower add 1 to attempts
        RTS

subby2
      SUB #$1,A
      JMP checki2
      RTS
```

```
*loop back to check for new digits AND ADD 1 TO ATTEMPTS

LOWER   MOVE B,NUMBER
        MOVE A,DUMP
lower   MOVE $E3,B *check if console output is ready
        CMP #$0,B
        BEQ lower
        MOVE #$76,A  *Move ASCII L TO A
        MOVE A,$E2   *WRITE TO CONSOLE/DISPLAYS
checki  CMP #$0,A
        BNE subby
        ADD #$1,A
        MOVE A,ATTEMPTS
        RTS

subby
      SUB #$1,A
      JMP checki


*loop back to check for new digits ADD 1 TO ATTEMPTS
****************************
******SHOW ATTEMPTS******
****************************
SATTEMPTS MOVE A,DUMP
          MOVE B,DUMP
S         MOVE $E3,B *check if console output is ready
          CMP #$0,B
          BEQ S *GO BACK
          MOVE ATTEMPTS,A
          MOVE A,$E2 *WRITE TO CONSOLE
          HALT *once attempts shown halt as correct answer given



****************************************
 *ADDING SYSTEM*
****************************************
*firstDigit*

addingOne
          MOVE A,INPUTS
comp      CMP #$0,A
          BNE take1
          ADD #$0a,A *ADD 10 TO A
          MOVE A,1stDigit *MOVES TEN TO 1ST DIGIT
          MOVE INPUTS,A
          RTS


take1  SUB #$1,A *TAKE 1 TILL EQUAL TO 0
       JMP comp



*****************************************
*secondDigit*
*1
adding   MOVE A,INPUTS *clear A
comp1    CMP #$0,A
         BNE take
         MOVE #$1,A
         MOVE A,2ndDigit
         MOVE INPUTS,A  *move inputs back
         RTS


take  SUB #$1,A
      JMP comp1



*2
adding1  MOVE A,INPUTS *clear A
comp2    CMP #$0,A
         BNE take2
         MOVE #$2,A
         MOVE A,2ndDigit
         MOVE INPUTS,A  *move inputs back
         RTS


take2  SUB #$1,A
       JMP comp2
```

```
*3
adding2  MOVE A,INPUTS *clear A
comp3    CMP #$0,A
         BNE take3
         MOVE #$3,A
         MOVE A,2ndDigit
         MOVE INPUTS,A  *move inputs back
         RTS


take3  SUB #$1,A
       JMP comp3


*4
adding3  MOVE A,INPUTS *clear A
comp4    CMP #$0,A
         BNE take4
         MOVE #$4,A
         MOVE A,2ndDigit
         MOVE INPUTS,A  *move inputs back
         RTS


take4  SUB #$1,A
       JMP comp4



*5

adding4  MOVE A,INPUTS *clear A
comp5    CMP #$0,A
         BNE take5
         MOVE #$5,A
         MOVE A,2ndDigit
         MOVE INPUTS,A  *move inputs back
         RTS


take5  SUB #$1,A
       JMP comp5


*6
adding5  MOVE A,INPUTS *clear A
comp6    CMP #$0,A
         BNE take6
         MOVE #$6,A
         MOVE A,2ndDigit
         MOVE INPUTS,A  *move inputs back
         RTS


take6  SUB #$1,A
       JMP comp6

*7
adding6  MOVE A,INPUTS *clear A
comp7    CMP #$0,A
         BNE take7
         MOVE #$7,A
         MOVE A,2ndDigit
         MOVE INPUTS,A  *move inputs back
         RTS


take7  SUB #$1,A
       JMP comp7

*8
adding7  MOVE A,INPUTS *clear A
comp8    CMP #$0,A
         BNE take8
         MOVE #$8,A
         MOVE A,2ndDigit
         MOVE INPUTS,A  *move inputs back
         RTS

take8  SUB #$1,A
       JMP comp8


*9
adding8  MOVE A,INPUTS *clear A
comp9    CMP #$0,A
```

```
         BNE take9
         MOVE #$9,A
         MOVE A,2ndDigit
         MOVE INPUTS,A  *move inputs back
         RTS


take9  SUB #$1,A
       JMP comp9

************************************************
*FINAL COMPARISON BETWEEN NUMBER AND USER NUMBER*

*IF CORRECT DISPLAY CORRECT*
*IF LESS THAN DISPLAY HIGHER*
*- COMPARE A,B
*- REMOVE EVERYTHING OUT OF BOTH REGS
*- COMPARE USER NUMBER TO RANDOM NUMBER
*IF BPL THEN SAY LOWER
*IF BMI THEN SAY HIGHER
*IF BEQ THEN SAY CORRECT + JSR ATTEMPTS SUB
*SHOW AMOUNT OF ATTEMPTS WITH DISPLAY
*ONCE GUESSED and wrong DO JSR firstdigit+second digit again and compare to random

*CREATE LOOP BACK TO GUESS AND RESPONSE TILL CORRECT
*IF MORE THAN DISPLAY LOWER*
*LINK TO ATTEMPS AND DISPLAY ATTEMPT VALUE VAR*

************************
*VARIABLES*
************************
ATTEMPTS DC.W $3 *number of attempts var
1stDigit DC.W $1 *1st digit
2ndDigit DC.W $2 * 2nd digit input
NUMBER DC.W $0 *random number to guess
INPUTS DC.W $4 *A reg dump var
BINPUTS DC.W $5 *B reg dump var
DUMP DC.W $6 * TRASH
```