NES 6502 Basics

Knox Game Design

July 2026

Basics

- Number guessing game
- 6502 assembly
- Concepts
  - Generate random number
  - Displaying text and variables to the screen
  - User input, handling controller button presses
  - Control statements (branches, conditionals, loops)
  - Comparing values (equal, greater, less)
  - Incrementing values
  - Game states

KNOX GAME DESIGN

6502 registers

- A – accumulator
- X – Index register
- Y – Index register
- SP – Stack pointer
- PC – program counter
- P – processor status
  - bit 0: Carry flag
  - bit 1: Zero flag
  - bit 2: Interrupt Disable
  - bit 3: Decimal Mode
  - bit 4: Break Command
  - bit 5: Always 1 for NES
  - bit 6: Overflow flag
  - bit 7: Negative flag

Frequently Used 6502 OpCodes

- LDA – load accumulator
- STA – store accumulator
- CMP – compare accumulator

- INX – increment X
- STX – store X register
- CPX – compare X register

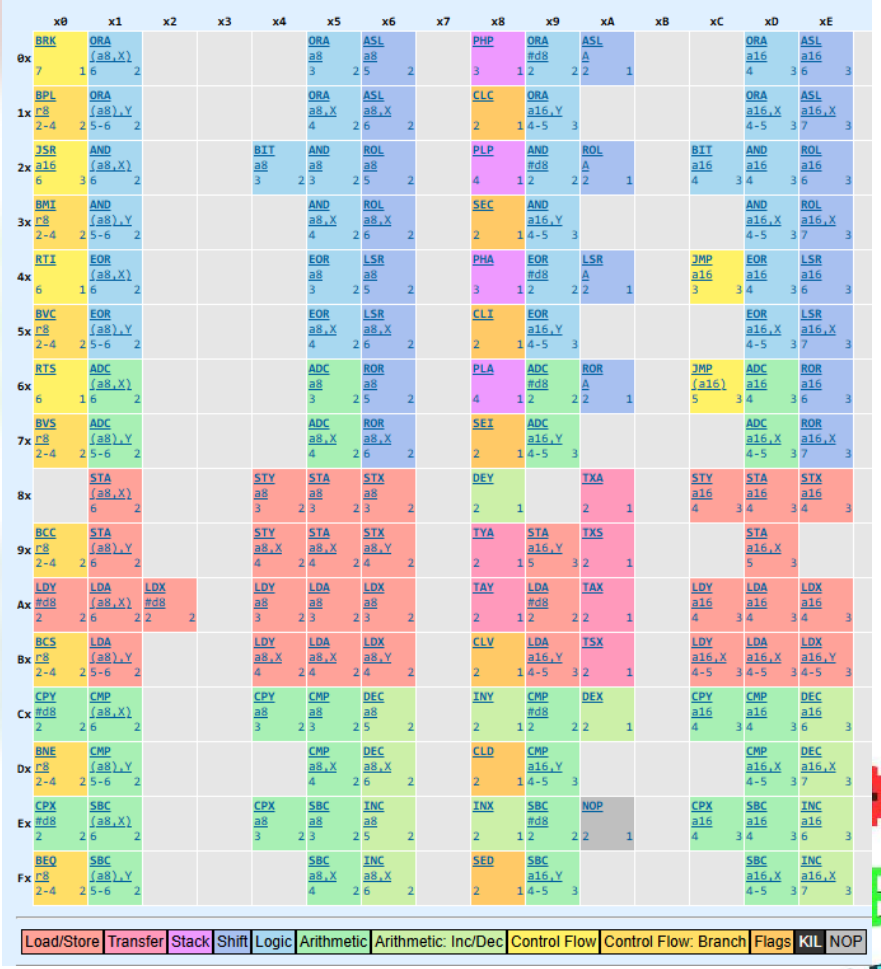
- SEI – set interrupt
- RTI – return from interrupt

- CLC – clear carry
- ADC – add with carry

- BEQ – branch on equal
- BNE – branch on not equal
- BCC – branch on carry clear
- BCS – branch on carry set

- JSR – jump to subroutine
- RTS – return from subroutine

- JMP - jump



source: pagetable.com

Generate Random Number

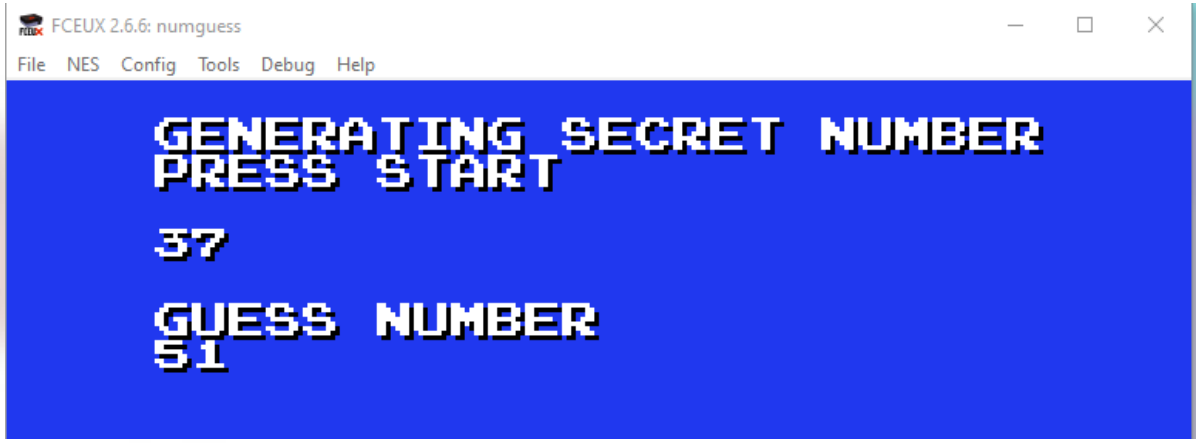
- Start a counter when program starts
- Increment on each update
- Stop when user presses start
- Options
  - Keep secret number stored as byte, then update the displayed digits
    - Must keep displayed values in sync with variable
  - Track digits (tens, ones) as separate values
    - Simpler to implement. The displayed values are the variables. No data redundancy
    - Wasted memory space. 8 bits (0 to 255) to hold a 4 bit (0 to 9) value
    - Digits roll over on increment 9


```
;continually increment two values 0 to 9
;until the user presses start
RandomizeSecret:
  LDA numsecret1
  CLC
  ADC #$01
  STA numsecret1

  CMP #$0a
  BNE RandomizeDone

  LDA #$00
  STA numsecret1
  LDA numsecret2
  CLC
  ADC #$01
  STA numsecret2

  CMP #$0a
  BNE RandomizeDone

  LDA #$00
  STA numsecret2

RandomizeDone:
  RTS
```

KNOX GAME DESIGN

Input Guessed Number

- Three states
  - Enter first digit
  - Enter second digit
  - Check answer
- Must track previous and current controller state
- Increment current digit value on D-pad up pressed on current and D-pad up released on previous
- Press “A” button to move to next state

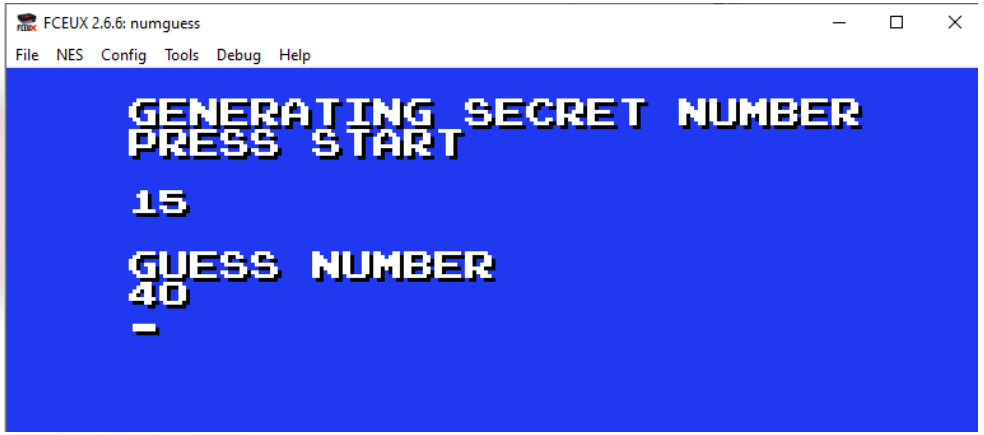


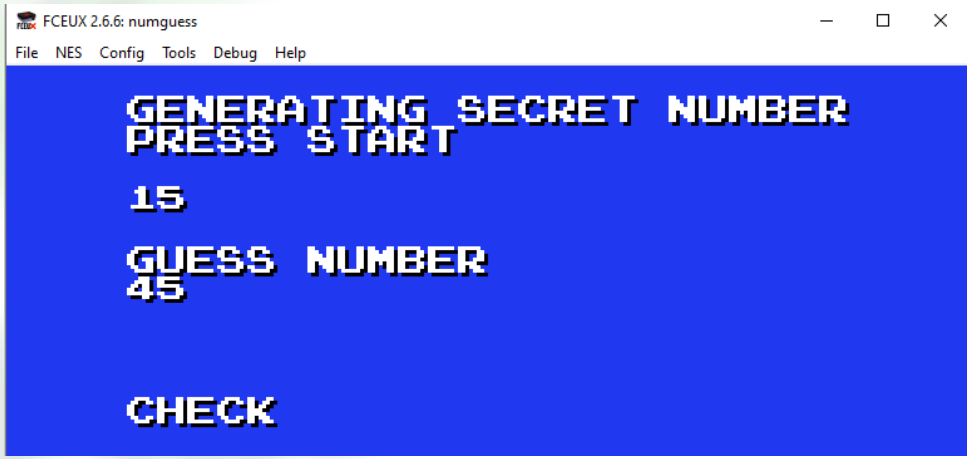

```
GameEngine:
  LDA gamestate
  CMP #$00
  BEQ EngineTitle

  LDA gamestate
  CMP #$01
  BEQ EngineInput1

  LDA gamestate
  CMP #$02
  BEQ EngineInput2

  LDA gamestate
  CMP #$03
  BEQ EngineInput3
GameEngineDone:
  RTI
```

KNOX GAME DESIGN

Add Cursor and Check

- Draw cursor based on current digit input
- Display message on number check
- Use str_to_nes to generate message strings





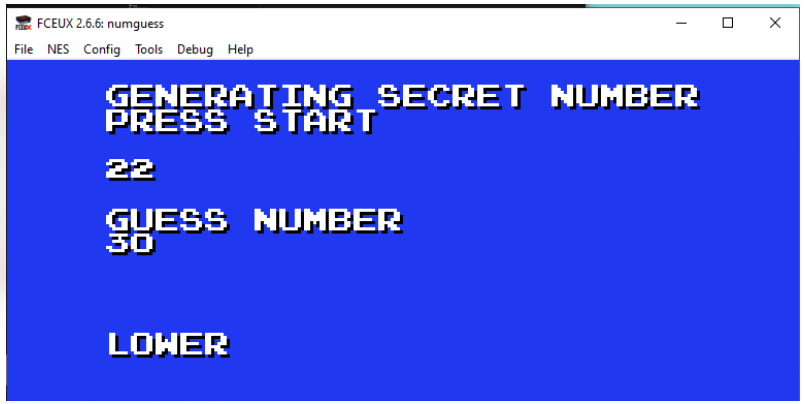

```
$ python str_to_nes.py
Enter string
check
CHECK
$0c,$11,$0e,$0c,$14
```

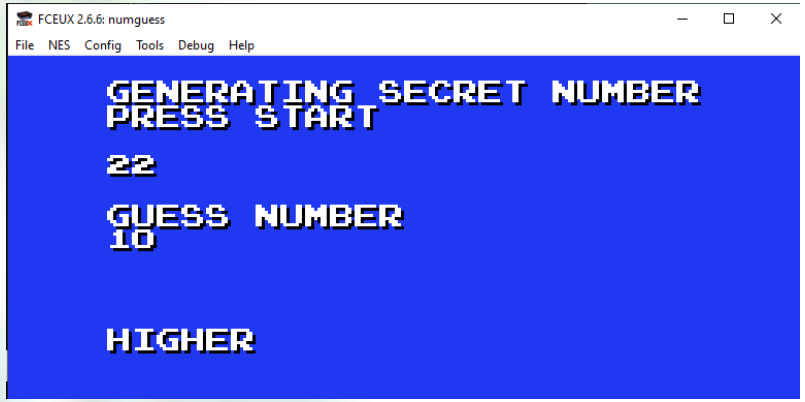
KNOX GAME DESIGN

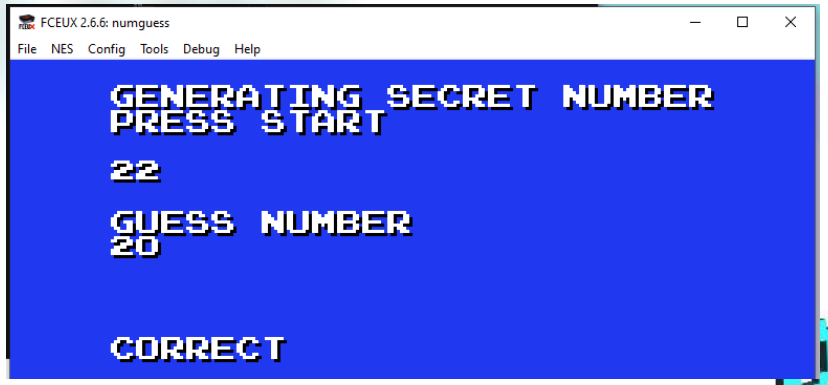
Check First (Tens) Digit

- Start by just checking one digit (tens digit)
- CMP – compare value with accumulator
- Branch to appropriate label
  - BEQ – equal
  - BCC – less than
  - BCS – greater than (or equal)

```
LDA numguess1
CMP numsecret1
BEQ DisplayCorrect
BCC DisplayHigher
BCS DisplayLower
```





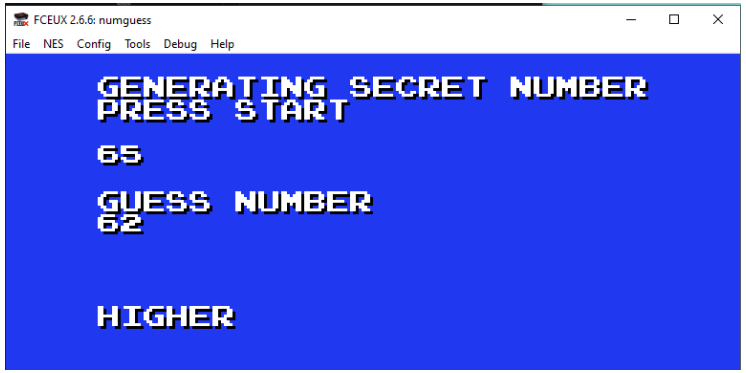


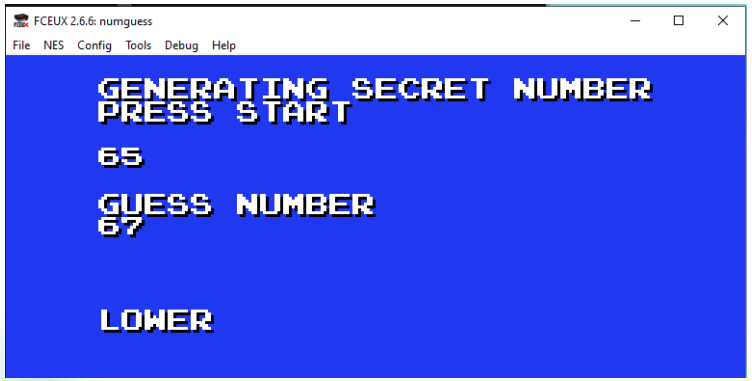
KNOX GAME DESIGN

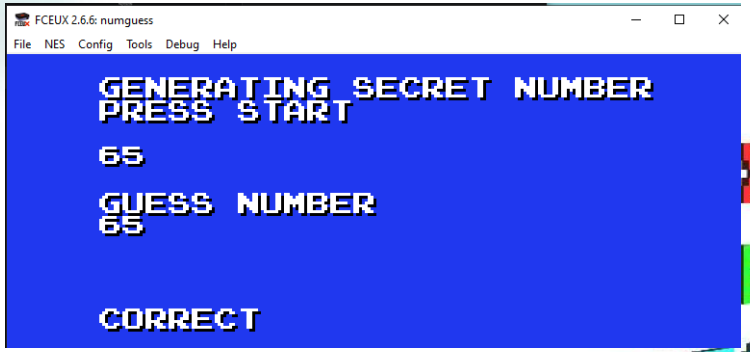
Check Second (Ones) Digit

- If the tens digit is equal, then check ones digit
- Ones digit check is similar to tens digit

```
CheckTensDigit:
  LDA gamestate
  CMP #$03
  BNE CheckOnesDigitDone

  LDA numguess1
  CMP numsecret1
  BEQ CheckOnesDigit
  BCC DisplayHigher
  BCS DisplayLower
CheckTensDigitDone:

CheckOnesDigit:
  LDA numguess2
  CMP numsecret2
  BEQ DisplayCorrect
  BCC DisplayHigher
  BCS DisplayLower
CheckOnesDigitDone:

  RTS
```

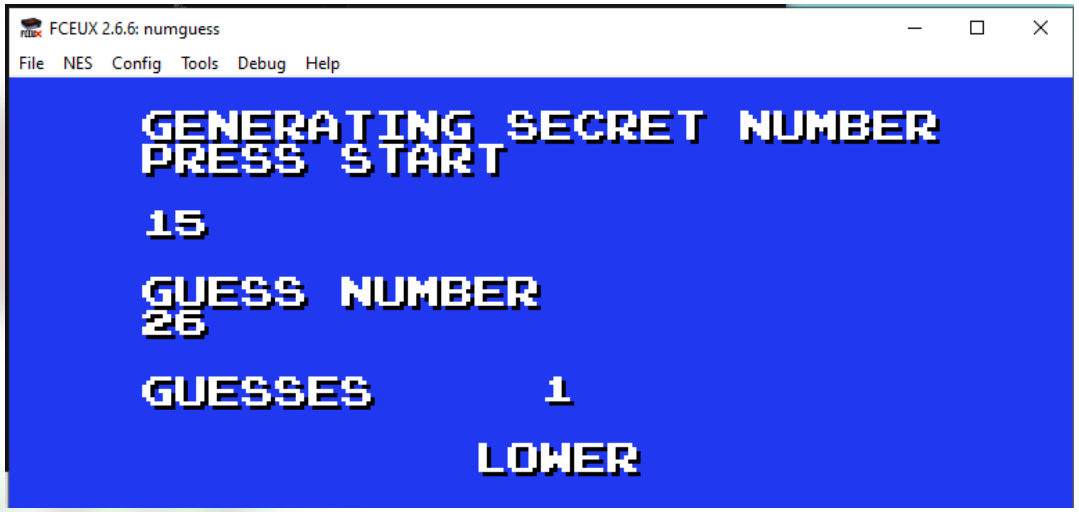






Add Guess Count

- Need an additional state so that guess check is only performed once
  - Otherwise guess will be continually checked. Guess count will continually increase
- Increment guess count if guess is incorrect
- New variable to hold result for display (0 = correct, 1 = higher, 2 = lower)




```
ResultCorrect:
  LDA #$00
  STA result

  LDA #$04
  STA gamestate
  JMP CheckDone
ResultCorrectDone:
```

```
ResultHigher:
  LDA #$01
  STA result

  JSR IncrementGuessCount
  LDA #$04
  STA gamestate
  JMP CheckDone
ResultHigherDone:

ResultLower:
  LDA #$02
  STA result

  JSR IncrementGuessCount
  LDA #$04
  STA gamestate
  JMP CheckDone
ResultLowerDone:
```

```
IncrementGuessCount:
  LDA numguesscount
  CLC
  ADC #$01
  STA numguesscount
IncrementGuessCountDone:
  RTS
```

KNOX GAME DESIGN

Game Completed

- When the correct number is guessed
  - Transition back to title state (generate new secret number)
  - Set guess count back to 0

KNOX GAME DESIGN