

The logo for Knox Game Design is positioned in the background. It consists of the word "KNOX" in red, "GAME" in green, and "DESIGN" in cyan, all in a pixelated, blocky font. The text is slightly blurred and has a subtle drop shadow.

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

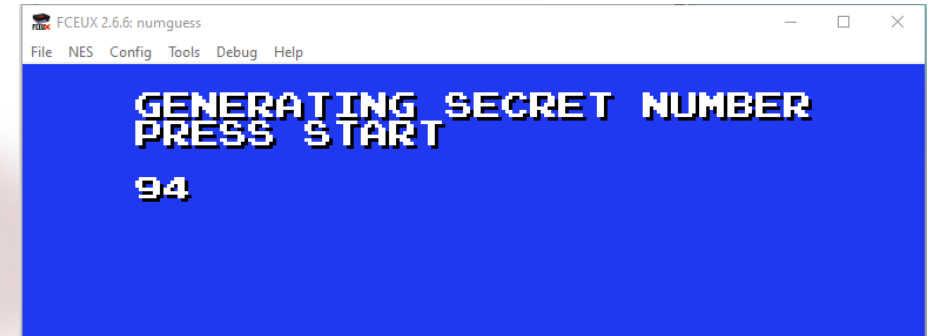
	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD	xE
0x	BRK 7 1 6 2	ORA {a8,X} 2				ORA a8 3 2 5 2	ASL a8 4 2 6 2		PHP 3 1 2 2 2 1	ORA md8 4 2 6 2	ASL A 4 2 6 2			ORA a16 4 3 6 3	ASL a16 4 3 6 3
1x	BPL 2-4 2 5-6 2	ORA {a8},Y 2				ORA a8,X 4 2 6 2	ASL a8,X 4 2 6 2		CLC 2 1 4-5 3	ORA a16,Y 4 2 6 2				ORA a16,X 4-5 3 7 3	ASL a16,X 4-5 3 7 3
2x	JSR a16 6 3 6 2	AND {a8,X} 2			BIT a8 3 2 3 2 5 2	AND a8 4 2 6 2	ROL a8 4 2 6 2		PLP 4 1 2 2 2 1	AND A 4 2 6 2	ROL A 4 2 6 2		BIT a16 4 3 4 3 6 3	AND a16 4 3 4 3 6 3	ROL a16 4 3 4 3 6 3
3x	BMI 2-4 2 5-6 2	AND {a8},Y 2				AND a8,X 4 2 6 2	ROL a8,X 4 2 6 2		SEC 2 1 4-5 3	AND a16,Y 4 2 6 2				AND a16,X 4-5 3 7 3	ROL a16,X 4-5 3 7 3
4x	RTI 6 1 6 2	EOR {a8,X} 2				EOR a8 3 2 5 2	LSR a8 4 2 6 2		PHA 3 1 2 2 2 1	EOR md8 4 2 6 2	LSR A 4 2 6 2		JMP a16 3 4 3 6 3	EOR a16 4 3 4 3 6 3	LSR a16 4 3 4 3 6 3
5x	BVC 2-4 2 5-6 2	EOR {a8},Y 2				EOR a8,X 4 2 6 2	LSR a8,X 4 2 6 2		CLI 2 1 4-5 3	EOR a16,Y 4 2 6 2				EOR a16,X 4-5 3 7 3	LSR a16,X 4-5 3 7 3
6x	RTS 6 1 6 2	ADC {a8,X} 2				ADC a8 3 2 5 2	ROR a8 4 2 6 2		PLA 4 1 2 2 2 1	ADC md8 4 2 6 2	ROR A 4 2 6 2		JMP {a16} 5 3 4 3 6 3	ADC a16 4 3 4 3 6 3	ROR a16 4 3 4 3 6 3
7x	BVS 2-4 2 5-6 2	ADC {a8},Y 2				ADC a8,X 4 2 6 2	ROR a8,X 4 2 6 2		SEI 2 1 4-5 3	ADC a16,Y 4 2 6 2				ADC a16,X 4-5 3 7 3	ROR a16,X 4-5 3 7 3
8x		STA {a8,X} 6 2			STY a8 3 2 3 2 3 2	STA a8 4 2 6 2	STX a8 4 2 6 2		DEY 2 1 4-5 3		TXA 2 1 4-5 3		STY a16 4 3 4 3 6 3	STA a16 4 3 4 3 6 3	STX a16 4 3 4 3 6 3
9x	BCC 2-4 2 5-6 2	STA {a8},Y 2			STY a8,X 4 2 6 2	STA a8,X 4 2 6 2	STX a8,Y 4 2 6 2		TYA 2 1 5 3 2 1	STA a16,Y 4 2 6 2				STA a16,X 5 3 4 3 6 3	
Ax	LDY a08 2 2 6 2	LDA {a8,X} 2	LDX md8 2 2 6 2		LDY a8 3 2 3 2 3 2	LDA a8 4 2 6 2	LDX a8 4 2 6 2		TAY 2 1 2 2 2 1	LDA md8 4 2 6 2	TAX 2 1 4-5 3		LDY a16 4 3 4 3 6 3	LDA a16 4 3 4 3 6 3	LDX a16 4 3 4 3 6 3
Bx	BCS 2-4 2 5-6 2	LDA {a8},Y 2			LDY a8,X 4 2 6 2	LDA a8,X 4 2 6 2	LDX a8,Y 4 2 6 2		CLV 2 1 4-5 3	LDA a16,Y 4 2 6 2	TSX 2 1 4-5 3		LDY a16,X 4-5 3 4-5 3	LDA a16,X 4-5 3 4-5 3	LDX a16,Y 4-5 3 4-5 3
Cx	CPY a08 2 2 6 2	CMP {a8,X} 2			CPY a8 3 2 3 2 5 2	CMP a8 4 2 6 2	DEC a8 4 2 6 2		INY 2 1 2 2 2 1	CMP md8 4 2 6 2	DEX 2 1 4-5 3		CPY a16 4 3 4 3 6 3	CMP a16 4 3 4 3 6 3	DEC a16 4 3 4 3 6 3
Dx	BNE 2-4 2 5-6 2	CMP {a8},Y 2				CMP a8,X 4 2 6 2	DEC a8,X 4 2 6 2		CLD 2 1 4-5 3	CMP a16,Y 4 2 6 2				CMP a16,X 4-5 3 7 3	DEC a16,X 4-5 3 7 3
Ex	CPX a08 2 2 6 2	SBC {a8,X} 2			CPX a8 3 2 3 2 5 2	SBC a8 4 2 6 2	INC a8 4 2 6 2		INX 2 1 2 2 2 1	SBC md8 4 2 6 2	NOP 2 1 4-5 3		CPX a16 4 3 4 3 6 3	SBC a16 4 3 4 3 6 3	INC a16 4 3 4 3 6 3
Fx	BEQ 2-4 2 5-6 2	SBC {a8},Y 2				SBC a8,X 4 2 6 2	INC a8,X 4 2 6 2		SED 2 1 4-5 3	SBC a16,Y 4 2 6 2				SBC a16,X 4-5 3 7 3	INC a16,X 4-5 3 7 3
	Load/Store	Transfer	Stack	Shift	Logic	Arithmetic	Arithmetic: Inc/Dec	Control Flow	Control Flow: Branch	Flags	KIL	NOP			

source: pagetable.com

DESIGN

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

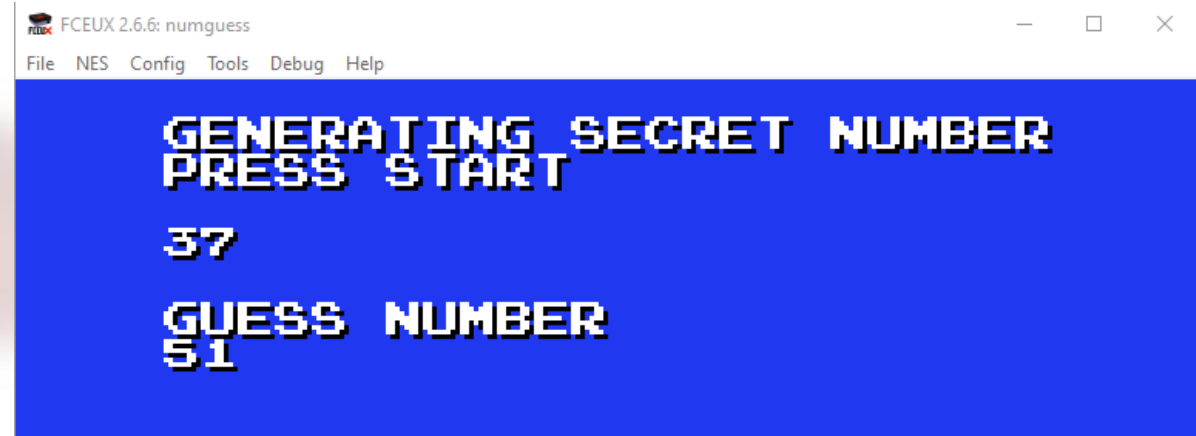
    LDA #$00
    STA numsecret2

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

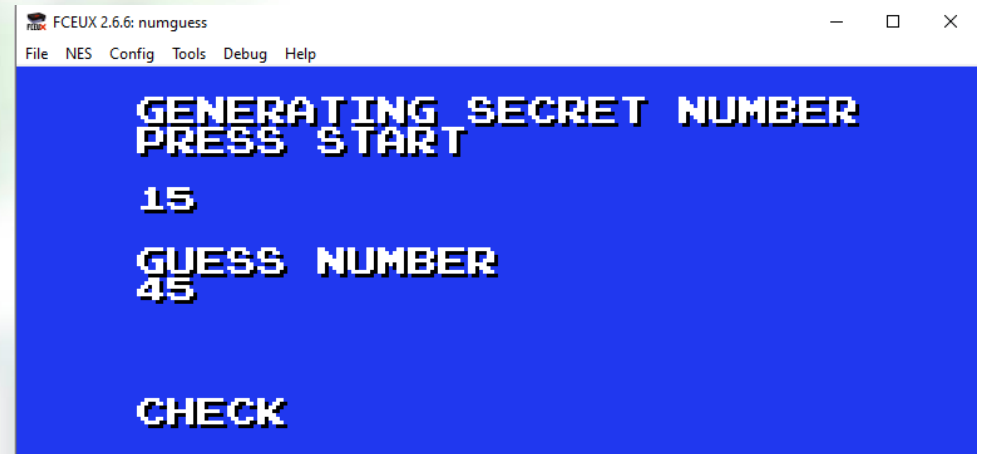
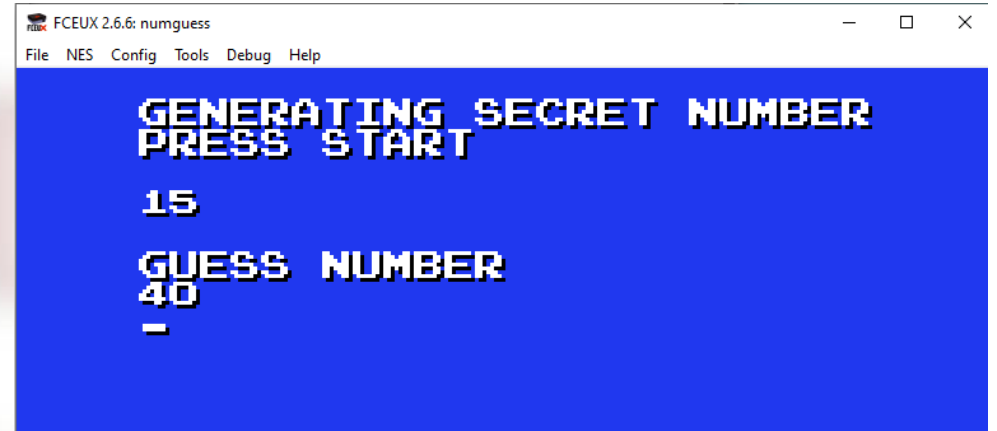
    LDA gamestate
    CMP #$03
    BEQ EngineInput3
GameEngineDone:
    RTI
```

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
```

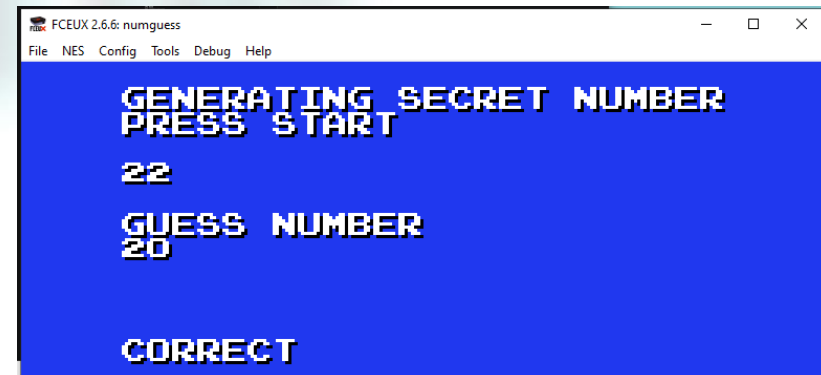
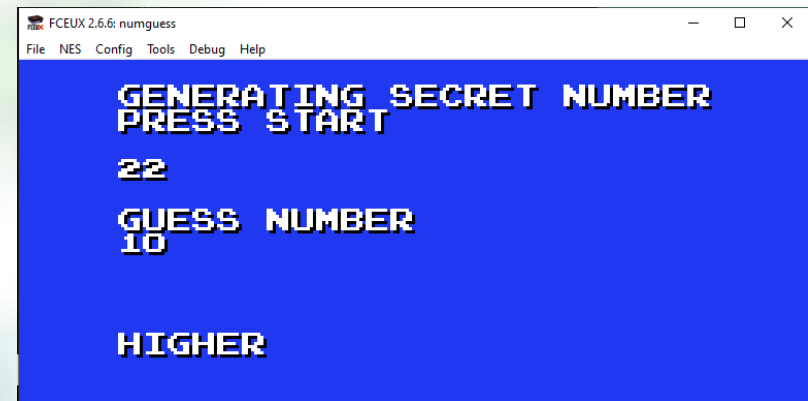
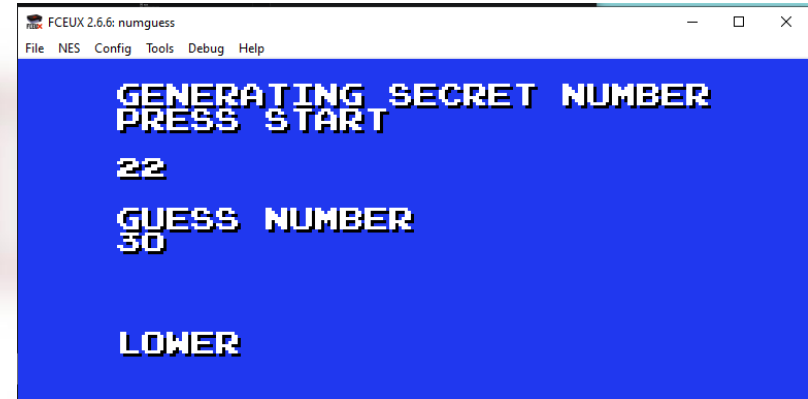


KNOX
GAME
DESIGN

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

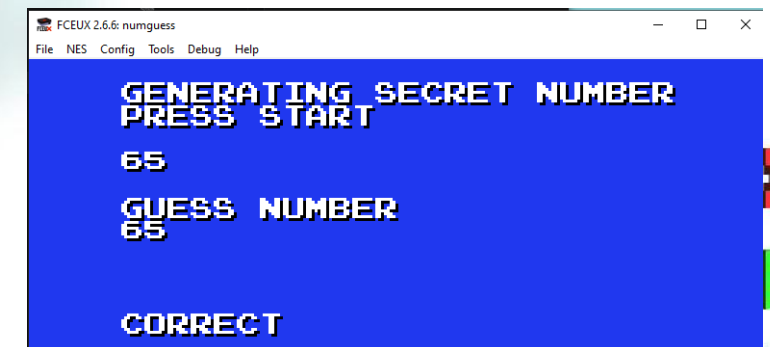
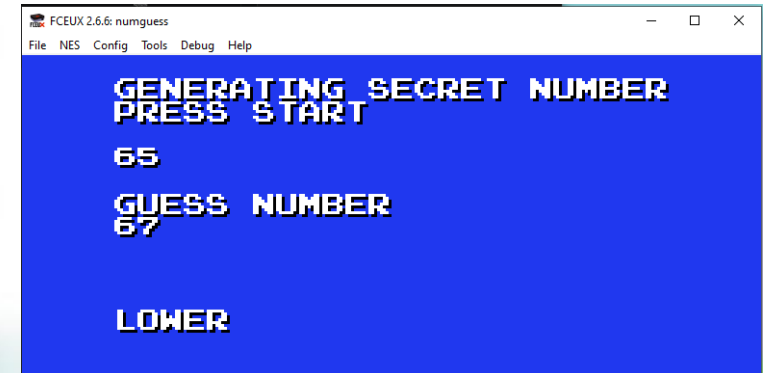
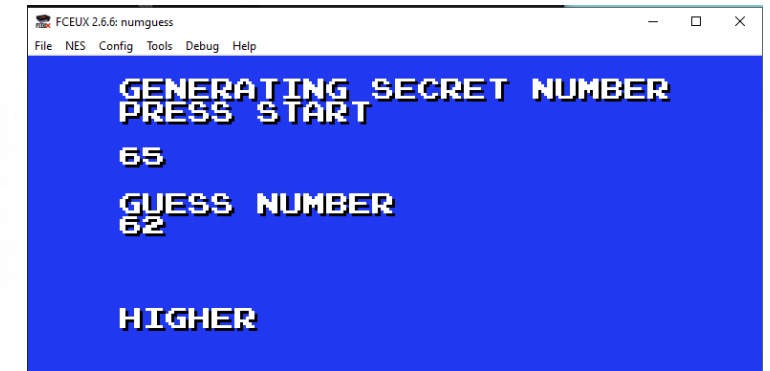
- If the tens digit is equal, then check ones digit
- Ones digit check is similar to tens digit

```
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:

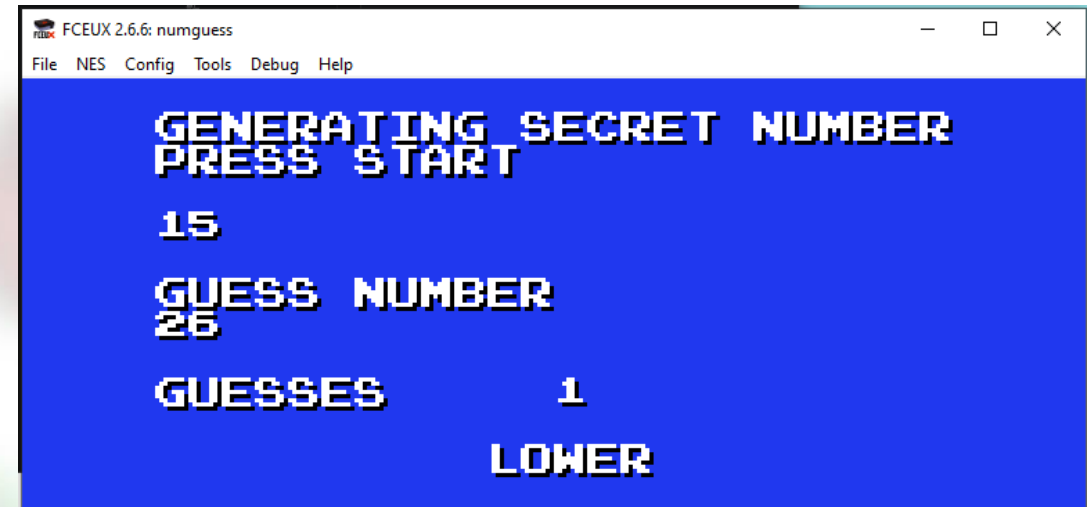
    RTS
```



DESIGN

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