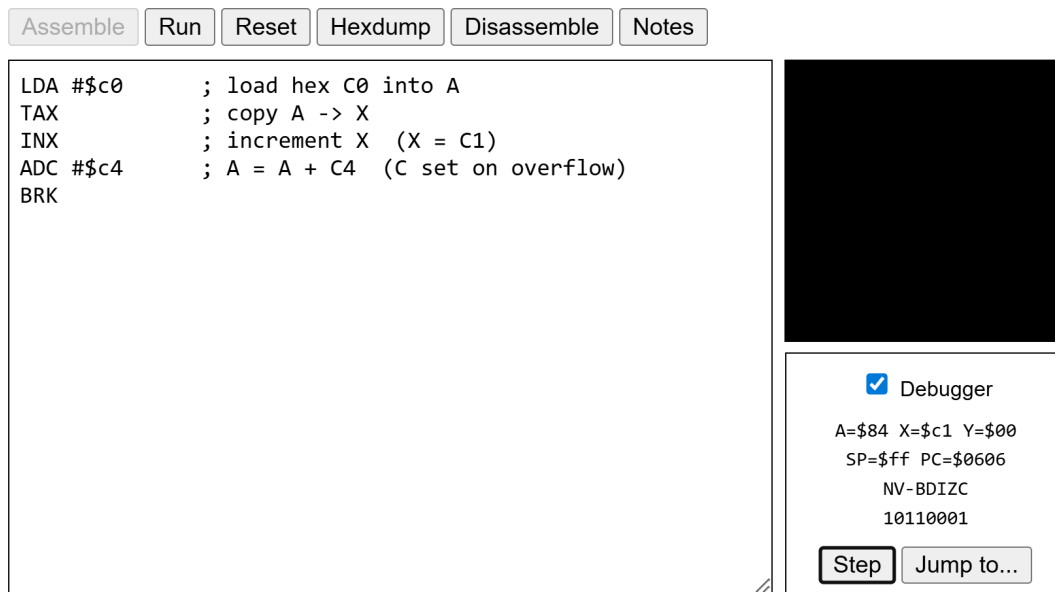


Lab Exercise 3: Instructions

Part 1

I used the top editor in easy6502. I assembled the starter program, turned on the debugger, and stepped until after the ADC instruction. I was instructed to skip the Companion Example in Part 1.

Step 6 result



The screenshot shows the easy6502 debugger interface. At the top are buttons for 'Assemble', 'Run', 'Reset', 'Hexdump', 'Disassemble', and 'Notes'. Below these is a text area containing assembly code:

```
LDA #$c0      ; load hex C0 into A
TAX           ; copy A -> X
INX           ; increment X (X = C1)
ADC #$c4      ; A = A + C4 (C set on overflow)
BRK
```

To the right of the code area is a black rectangle representing the memory dump. Below the code area is a box containing the debugger status:

☒ Debugger
A=\$84 X=\$c1 Y=\$00
SP=\$ff PC=\$0606
NV-BDIZC
10110001

At the bottom right of the debugger box are two buttons: 'Step' and 'Jump to...'.

Part 1 — Step 6: Post-ADC state showing A = \$84 and C = 1.

My variation

I changed the program so the low byte of the sum is in register Y and the carry is in register A. I assembled and stepped to confirm the result.

AssembleRunResetHexdumpDisassembleNotes

```
LDA #$c0      ; A = C0
CLC           ; clear carry before ADC
ADC #$c4      ; A = 84, Carry=1 (because C0 + C4 > FF)
TAY          ; Y = low byte (84)
LDA #$00      ; zero A WITHOUT touching Carry
ROL A        ; rotate carry into LSB of A => A=01 if
carry=1 else 00
BRK
```

☒ Debugger

A=\$01 X=\$00 Y=\$84
SP=\$ff PC=\$0609
NV-BDIZC
00110000

StepJump to...

Part 1 — Variation: Y holds the low byte and A reflects the carry (A = \$01).

Part 2

For Part 2 I used the lower editor. I turned on Monitor and Debugger, assembled, and stepped as needed. I captured each result as a screenshot.

Steps 7 to 9

The screenshot shows a 6502 assembler/debugger interface. At the top are buttons: Assemble, Run, Reset, Hexdump, Disassemble, and Notes. The main assembly window contains the following code:

```
LDA #$80
STA $01
ADC $01
```

To the right is a debugger window with the following information:

- ☒ Debugger
- A=\$00 X=\$00 Y=\$00
- SP=\$ff PC=\$0606
- NV-BDIZC
- 01110011
- Buttons: Step, Jump to...

Below the assembly window is a Monitor section with a checked checkbox, a Start address of \$0, and a Length of \$ff. Below this is a memory dump showing the first 100 bytes of memory (addresses 0000 to 0050) in hexadecimal:

```
0000: 00 80 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0010: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0020: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0040: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0050: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Part 2 — Steps 7 to 9: After ADC \$01. A = \$00, C = 1, Z = 1.

Another example with FF

Displaying the correct sum

Assemble
Run
Reset
Hexdump
Disassemble
Notes

```
LDA #$ff
STA $01
ADC $01
```

☒ Debugger

A=\$fe X=\$00 Y=\$00
SP=\$ff PC=\$0606
NV-BDIZC
10110001

Step
Jump to...

Monitor ☒ Start: \$ 0 Length: \$ ff

```
00a0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00b0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00c0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00d0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00e0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00f0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ec
```

Part 2 — Another example with \$FF: After ADC \$01. A = \$FE, C = 1, N = 1.

Assemble
Run
Reset
Hexdump
Disassemble
Notes

```

LDA #$85
CLC
ADC #$85
TAY
LDA #$00
ROL A
TAX
BRK

```

☒ Debugger

A=\$01 X=\$01 Y=\$0a
SP=\$ff PC=\$060a
NV-BDIZC
01110000

Step
Jump to...

Monitor ☒
Start: \$ 0
Length: \$ ff

```

00a0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00b0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00c0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00d0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00e0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00f0: 00 00 00 00 00 00 00 00 00 00 00 00 00 27

```

Part 2 — Displaying the correct sum for \$85 + \$85: Y = \$0A, A = \$01, X = \$01.

Step 10 answers

Q1 Can I run it after removing the two \$ signs?

A No. It does not assemble.

Q2 What happened?

A The assembler rejects the values without the \$ prefix.

Q3 Conclusion

A The \$ prefix is required for hexadecimal. Without it the values must be decimal.

Q4 Values needed to assemble and reproduce the original results

A Use decimal 192 and 196.

Step 11

I created a branching version that uses register Y as the counter. I will submit this program as a separate text file named Bruce Lab 3 Step11.txt.