nesdoug

30. ASM part 5

Probably the final 6502 ASM lesson. I'm going to try to cover everything I forgot.

Switch (foo){
case 0:
…
break;
case 1:
…
break;
case 2:
…
}

Let's say we have a variable 'state' that if state = 0, we do one thing. If state = 1, we do another thing. Etc. Generally, it would be handled like this…

```
    LDA foo      load A from address foo, will set a zero flag if foo = 0
    BNE Check_1  branch ahead if foo != 0
    ...
    JMP Done     break



Check_1:         A is still loaded with value from foo
    CMP #1       compare to value 1, sets zero flag if foo = 1
    BNE Check_2  branch ahead if foo != 1
    ...
    JMP Done     break



Check_2:         A is still loaded with value from foo
    CMP #2       compare to value 2, sets zero flag if foo = 2
    BNE Done     branch ahead if foo != 2
    ...
```

```
Done:
```

Comments are done with a ; in ca65

;this is a comment

You add additional ASM source code like this…

.include “Second_ASM_File.asm”

You add binary files like this…

.incbin “Something.bin”

Often, you put a binary file just below a label, so you can index it from the label.

Label_Name:
.incbin “Something.bin”

You might wonder why I never add the lines .P02 (to set to 6502 mode) or -t nes (to set the target as ‘NES’), and the answer is…it makes no difference. The default mode of ca65 assembles exactly how I want, so I don’t bother.

I skipped over the V flag (overflow). The V flag is only set by ADC and SBC (and BIT). This is a way to treat the numbers as signed -128 to +127.

For ADC, the V flag is only set if 2 positive numbers add together to get a negative, or if 2 negative numbers add together to get a positive. Any other ADC operation will clear the V flag.

For SBC, the V flag is only set if Pos-Neg = Neg…or if Neg-Pos = Pos. All other SBC operations will clear the V flag.

What you do with the result is up to you, but you have BVC (branch if V clear) and BVS (branch if V set) to help you decide.

MULTIPLICATION/DIVISION

I went and wrote several routines that would do these as efficiently as I could think…and then I found these webpages, which do the same thing about 10x faster.

http://6502org.wikidot.com/software-math-intmul (http://6502org.wikidot.com/software-math-intmul)

http://6502org.wikidot.com/software-math-intdiv (http://6502org.wikidot.com/software-math-intdiv)

I’ve tried them out. They work great.

OH, and before I forget, I found another webpage with an online assembler, that you can test out code.

https://skilldrick.github.io/easy6502/ (https://skilldrick.github.io/easy6502/)

I can't think of anything else at this time, but you can look at these resources for more infomation…

http://www.6502.org/ (http://www.6502.org/)

http://wiki.nesdev.com/w/index.php/Programming_guide
(http://wiki.nesdev.com/w/index.php/Programming_guide)

http://wiki.nesdev.com/w/images/7/76/Programmanual.pdf
(http://wiki.nesdev.com/w/images/7/76/Programmanual.pdf)

The last one has lots of info about 65C02 and 65816 processors too, but if you scroll down to Chapter 18 (p.326) it will describe how all the instructions work. Most of them are relevant to 6502 programming also.

March 13, 2016April 13, 2017 dougfraker