codeBrane Blog

codebrane.com/blog/2013/10/25/hello-world-in-commodore-64-assembly/

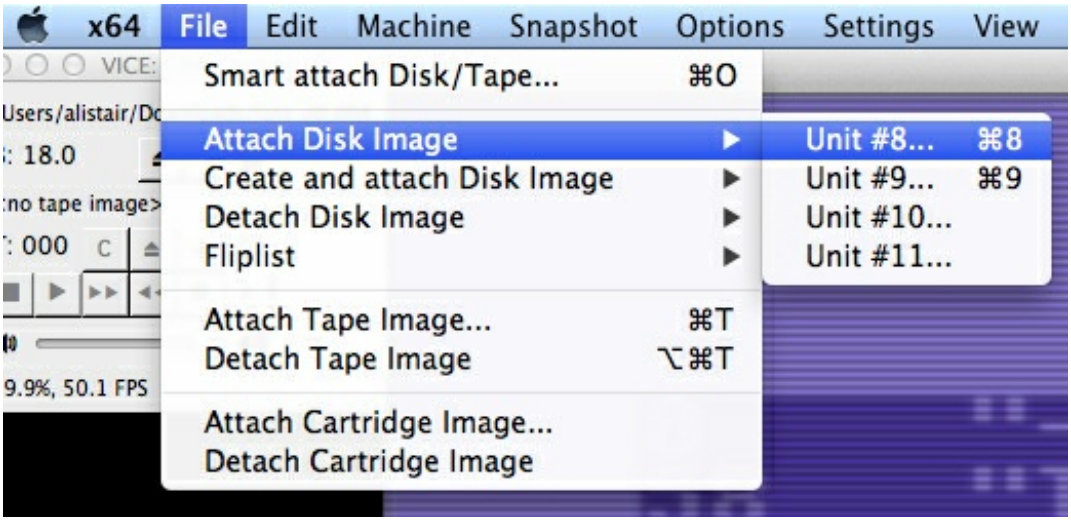
I've been rebooting my interest in the Commodore 64 recently, after a layoff of around 20 years! The last time I used a 64 was back in 1992 when I wrote a BASIC word processor on which I typed my honours thesis and printed out on an MPS printer, if I remember the model correctly.

I still have the old machine, printer, tape drive etc but I've been relearning everything on VICE before I get round to powering them up again. So this is a wee tutorial on how to display a simple Hello World message in C64 assembler. It won't be 'type and run' sort of thing as I'll show how to add a BASIC header so you can compile the object code and run it as a normal program from BASIC. So, let's crack on.

First go and get the VICE emulator. I'm on a Mac so I downloaded it from here. Before you start it up, download the TASM assembler from here. Unzip to get a disk image file:

> *tassdemo.d64*

Now start up VICE using the x64 binary and attach tassdemo.d64 to Unit 8.



If you want to see what's on the disk, type this:

> *LOAD "$",8*
> *LIST*

It's interesting to see what's available but from now on, the quickest way to load up the assembler is to type:

> *LOAD "TURBO ASS MAC+",8,1*

but before you hit the enter key, max up VICE by choosing:

> *Options -> Maximum Speed -> No Limit*

now hit enter and it will load really quickly. Then set the speed back to normal:

> *Options -> Maximum Speed -> 100%*

To run the assembler, type:

> *sys 8*4096*

this tells the CPU to start working at that specific memory address, which is where tasm is loaded. So you should now be looking at the weirdly blank tasm screen. Welcome to a 1980s development environment!

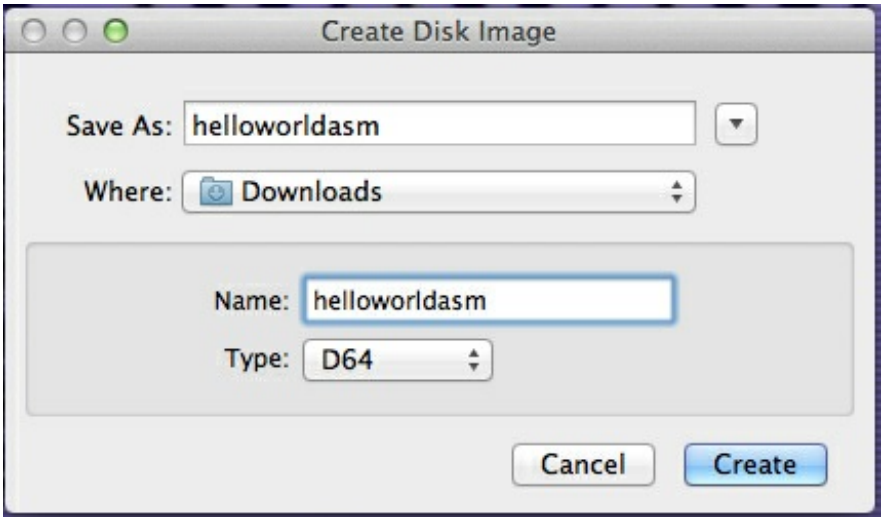
The first thing we need to do is prepare our local environment to save our work. So, detach the tasm disk as we no longer need it. The program is running from memory now:

> *File -> Detach Disk Image -> Unit #8*

and create and attach a new disk image:

> *File -> Create and attach Disk Image -> Unit #8*

this is what mine looked like:



Now you have what is basically an empty disk attached to your C64 where you can save programs.

Normally in assembler tutorials you start off by putting the code at a specific address and coding away. Then you assemble and run the program in the assembler itself. But we're not going to do that. We're going assemble the program and test it in the assembler but then we're going to assemble it to disk and run it from BASIC. As if it was a program we'd bought on tape. So how do you do that?

On the Commodore 64, BASIC memory starts at 2048 (0x800) so we'll put some code in there which the C64 will interpret as a BASIC program. It'll be a one liner:

> *10 sys 4096*

When you want to run a machine language program you use the sys command, as we did when running tasm. 4096 is the decimal equivalent of 0x1000 which is where we'll start our assembly code. So the top of the file will look a bit messy:

```
*=$0801
.byte $0c, $08, $0a, $00, $9e, $20
.byte $34, $30, $39, $36, $00, $00
.byte $00
```

All this does is let you do this when you load the program:

> *LIST*
> *10 SYS 4096*
> *RUN*

rather than having to resort to sys yourself. So, on to the actual program itself.

```
*=$0801
.byte $0c, $08, $0a, $00, $9e, $20
.byte $34, $30, $39, $36, $00, $00
.byte $00
*=$1000
ldx #$000
loop lda message,x
and #$3f
sta $0400,x -> 1024
inx
cpx #$0c
bne loop
rts
message
    .text "Hello World!"
```

What's going on?

We're loading the program into memory at 0x1000, 4096 decimal (hence the SYS 4096 from BASIC) and the first thing we do is zero the X register:

> *ldx #$000*

we then have a bit of code that has a label (loop). This lets us reference this line of code later and the code just loads the address of message plus an offset of what's in the X register, and stores the value in the accumulator (what a wonderful word!). So the first time through the loop X will be 0 and we'll get the first byte of message, i.e. 'H':

> *loop lda message,x*

The next line is just plain weird. If we didn't do this we'd get garbage output unless we were in lowercase mode in BASIC. The reason is to do with ASCII/PETSCII and this is the easiest way to deal with it. Convert to UPPERCASE!

> *and #$3f*

We now need to output to the screen. We want to do this ourselves rather than get a kernel interrupt to do it. Maybe that's too modern a term, interrupt! Screen memory starts at 1024 decicmal, or 0x400. What we do here is store the value in the accumulator at address 0x400 plus an offset determined by the value of the X register, which is currently zero as this is the first time through the loop. So the first letter will be displayed in the top left of the screen.

> *sta $0400,x*

Next is easy, increment the value in the X register:

> *inx*

and compare the result with 0x0c which is decimal 12, the length of the message:

> *cpx #$0c*

now we say ‘branch if not equal’ to the label called ‘loop’:

> *bne loop*

and we’ll go through the loop another time. This time loading and displaying ‘e’ from message. When the loop terminates we just return from the program:

> *rts*

and that’s it!

In the tasm assembler, in order to do things other than type code, we need to use the command key. This is the key to the left of the ‘1’ key on our keyboard. On my MacBook it’s the key with two symbols on it. The bottom one is something weird and coily with a stacked +/- above it. Above that key is the ‘esc’ key but the ‘weird coily stacked +/- key’ is the tasm command key. Do this, where the second key is pressed after the COMMAND key and not at the same time:

> *COMMAND 3*

this will assemble the program. Press any key to return to tasm. Now save your code. You should still have the new disk you created, attached on Unit 8:

> *COMMAND w*

choose a filename and hit enter. I chose ‘hello.asm’ (without the quotes). Now compile an object file to the same disk image:

> *COMMAND 5*

choose a filename of ‘hello’ (without the quotes). You should verify it’s saved ok by listing the contents of the disk directory:

> *COMMAND **

mine looks like this:

At this point we’ve finished with assembly so just hard reset or turn off/on the emulator.

Power up the emulator and attach your new disk image to Unit 8 and load it up. You should see your program in its code and compiled forms:

> *LOAD “$”,8*
> *LIST*
> *2 “HELLO.ASM” SEQ*
> *9 “HELLO” PRG*

just load up the HELLO program:

> *LOAD “HELLO”,8,1*

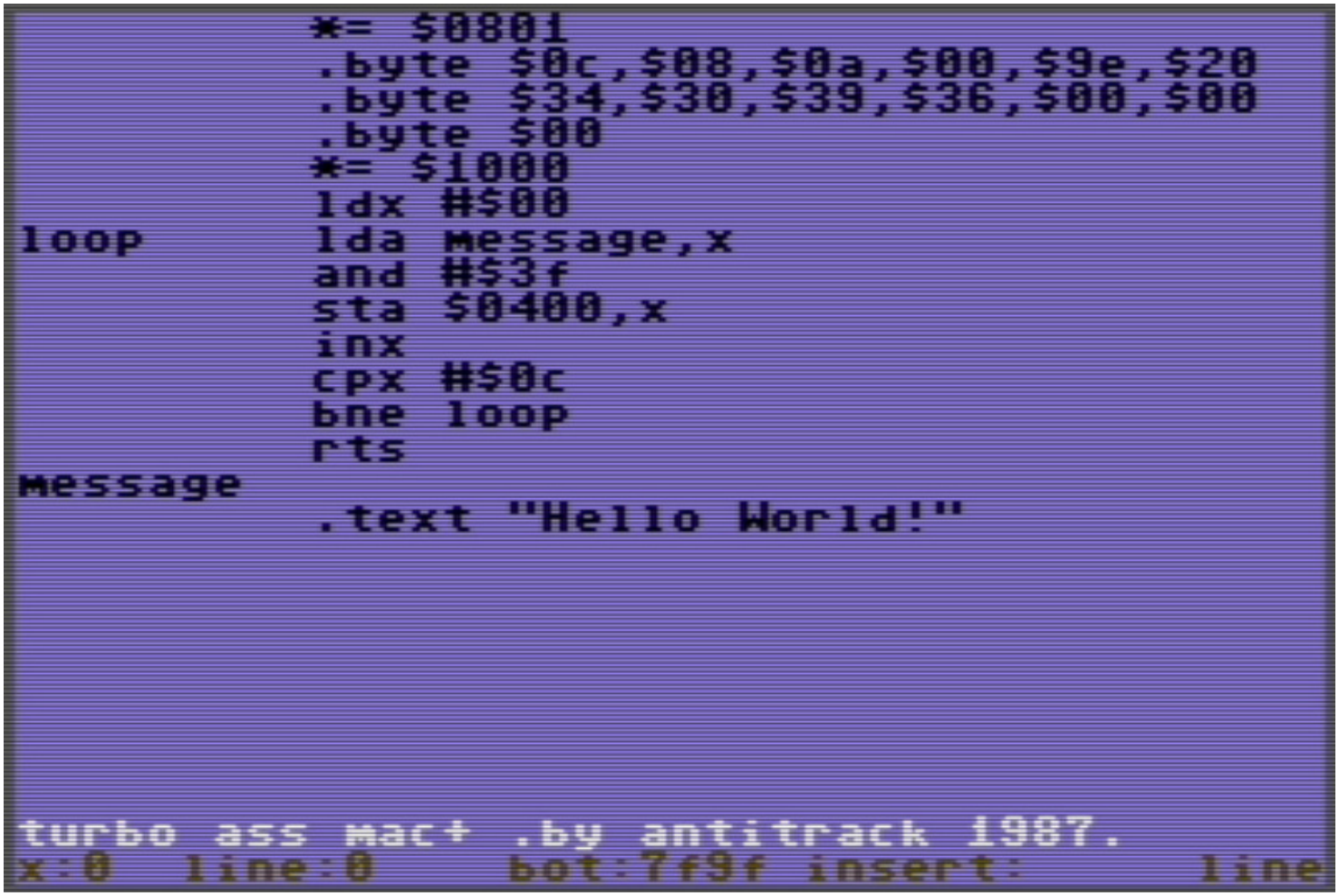
We haven’t written assembly code to clear the screen so let’s do that manually:

> *PRINT CHR$(147)*

and now we can run our program:

> *RUN*

voila! One assembly language program running on your Commodore 64!


*= $0801
.byte $0c,$08,$0a,$00,$9e,$20
.byte $34,$30,$39,$36,$00,$00
.byte $00
*= $1000
ldx #$00
loop lda message,x
and #$3f
sta $0400,x
inx
cpx #$0c
bne loop
rts
message
.text "Hello World!"
turbo ass mac+ .by antitrack 1987.
x:0 line:0 bot:7f9f insert: line

References: