HowToCode7

COLLABORATORS			
	TITLE : HowToCode7		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		August 9, 2024	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

HowToCode7

1.1 HowToCode: Beginners

```
How do I begin?
---------------

If you are just starting to learn programming, and you want
a good place to begin learning assembler, buy Amos!. It's
very easy to write assembler code, load it into amos and test it, and
if you end up being hopeless with assembler, you can write your
demos in AMOS instead! :-)

For example, take this routine:

;simplemaths.s

        add.l  d0,d1    ; add contents of d0 to d1
        rts

Assemble this with Devpac and what do you get? Not a lot.

Now, load AMOS and type this:

Pload "ram:simplemaths",1  ' load executable file into bank 1
Input "Enter a number ";n1
Input "Enter another number ";n2
dreg(0) = n1               ' Store n1 in 68000 register d0
dreg(1) = n2               ' Store n2 in 68000 register d1
call(1)                    ' Run your machinecode routine
Print n1;" plus ";n2;" equals ";dreg(1)    ' returns result in d1

You can start playing with 68000 instructions this way, seeing how
they work, without having to 'jump in the deep end' writing
routines to set up displays, copperlists, windows or writing to
the console.

You can also pass your machine code the address of AMOS's
bitplanes (by Phybase(0) to Phybase(n) where n is number of
bitplanes - 1), so you can write your own vector/bob code
and test it easily before writing your own front end code.
```

Once you have got the hang of 68000, you can drop Amos.

Another good way is to write some code in C, and use the inline debugging options with SAS C, and OMD to examine what your C compiler actually generates. To do this with SAS V6.x do the following

```
SC debug=full myprog.c

OMD >ram:omdoutput myprog.o myprog.c
```

You will get each line of C code interleaved with the assembler that it generates. Very handy!

It’s also amazing how good the code generated by SAS C 6.2 really is.