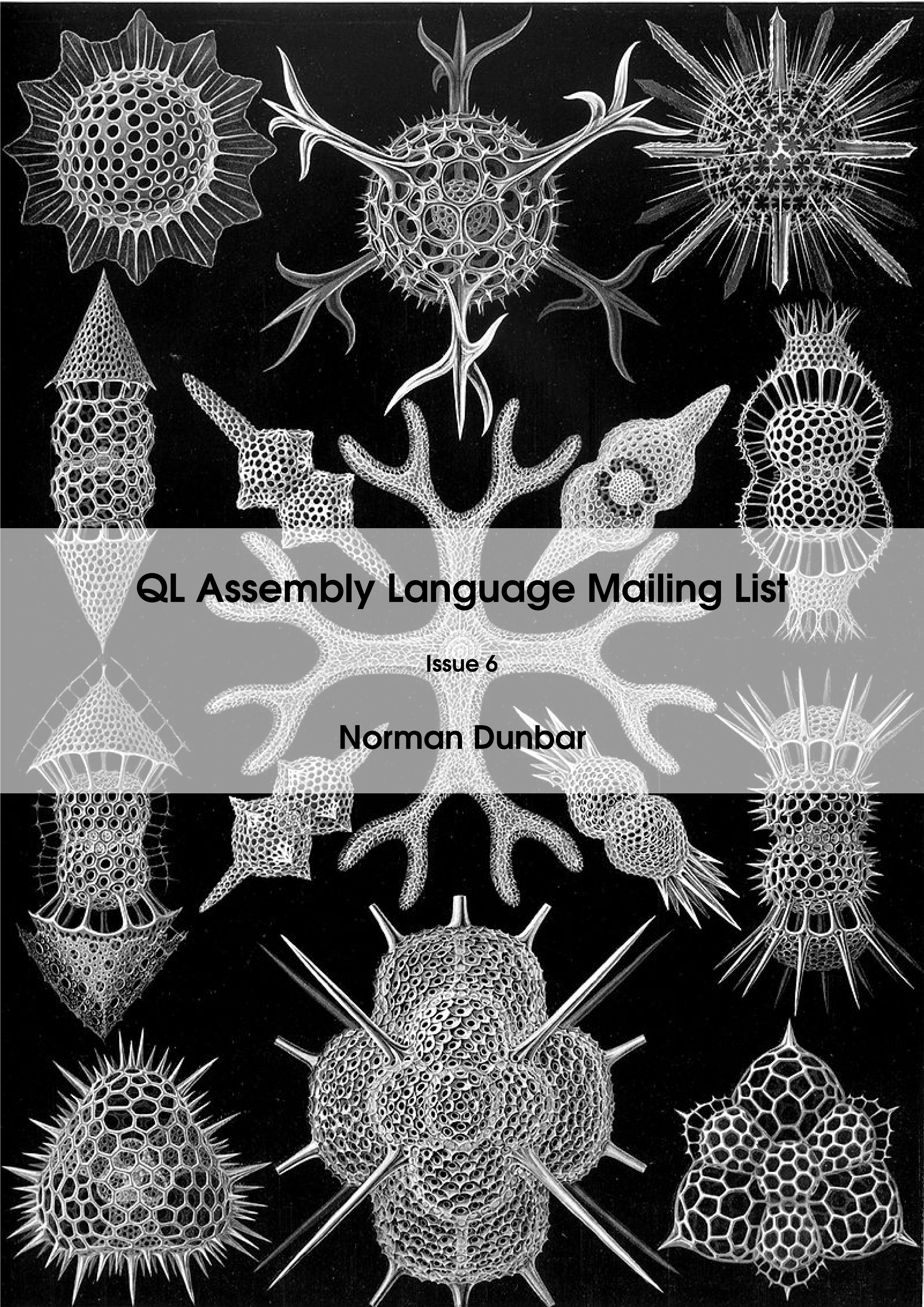


QL Assembly Language Mailing List

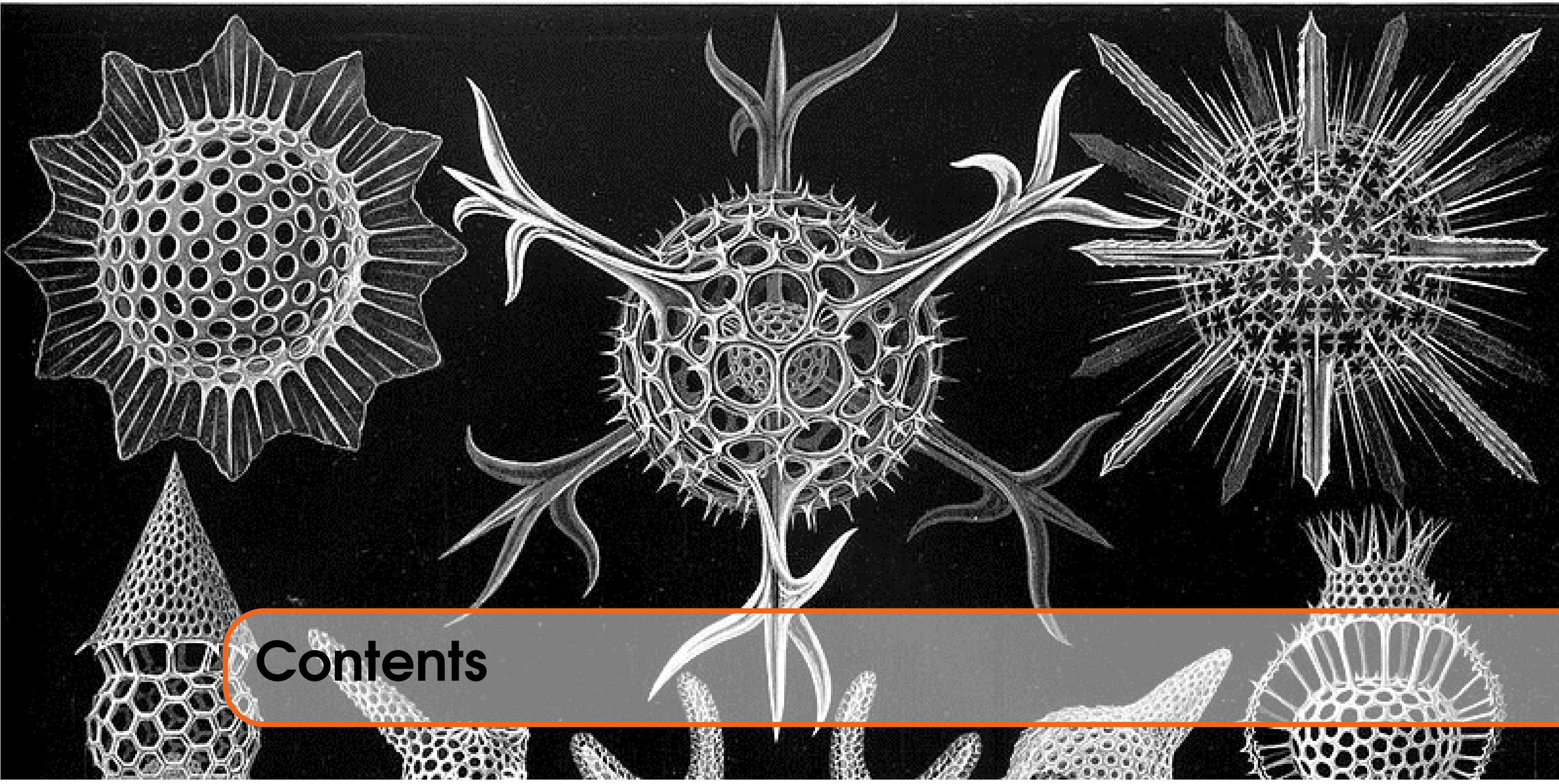
Issue 6

Norman Dunbar

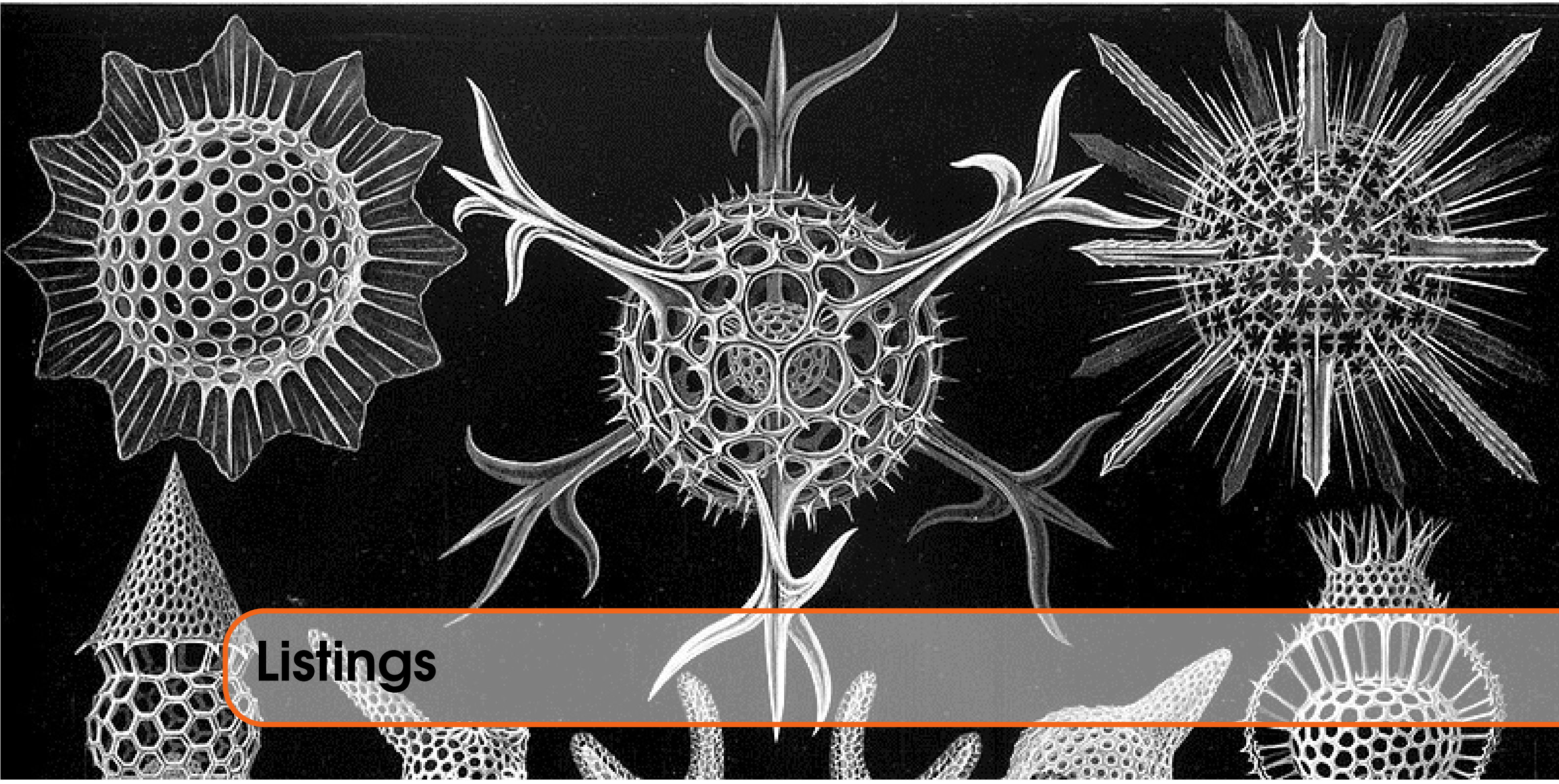
PUBLISHED BY MEMYSELFEYE PUBLISHING ;-)

http://qdosmsq.dunbar-it.co.uk/downloads/AssemblyLanguage/Issue_006/Assembly_Language_006.pdf

This pdf document was created on *1/3/2018* at *17:07:05*.



Contents



Listings

1. Preface

1.1 Feedback

Please send all feedback to assembly@qdosmsq.dunbar-it.co.uk. You may also send articles to this address, however, please note that anything sent to this email address may be used in a future issue of the eMagazine. Please mark your email clearly if you do not wish this to happen.

This eMagazine is created in LaTeXsource format, aka plain text with a few formatting commands thrown in for good measure, so I can cope with almost any format you might want to send me. As long as I can get plain text out of it, I can convert it to a suitable source format with reasonable ease.

I use a Linux system to generate this eMagazine so I can read most, if not all, Word or MS Office documents, Quill, Plain text, email etc formats. Text87 might be a problem though!

1.2 Subscribing to The Mailing List

This eMagazine is available by subscribing to the mailing list. You do this by sending your favourite browser to http://qdosmsq.dunbar-it.co.uk/mailinglist and clicking on the link “Subscribe to our Newsletters”.

On the next screen, you are invited to enter your email address *twice*, and your name. If you wish to receive emails from the mailing list in HTML format then tick the box that offers you that option. Click the Subscribe button.

An email will be sent to you with a link that you must click on to confirm your subscription. Once done, that is all you need to do. The rest is up to me!

1.3 Contacting The Mailing List

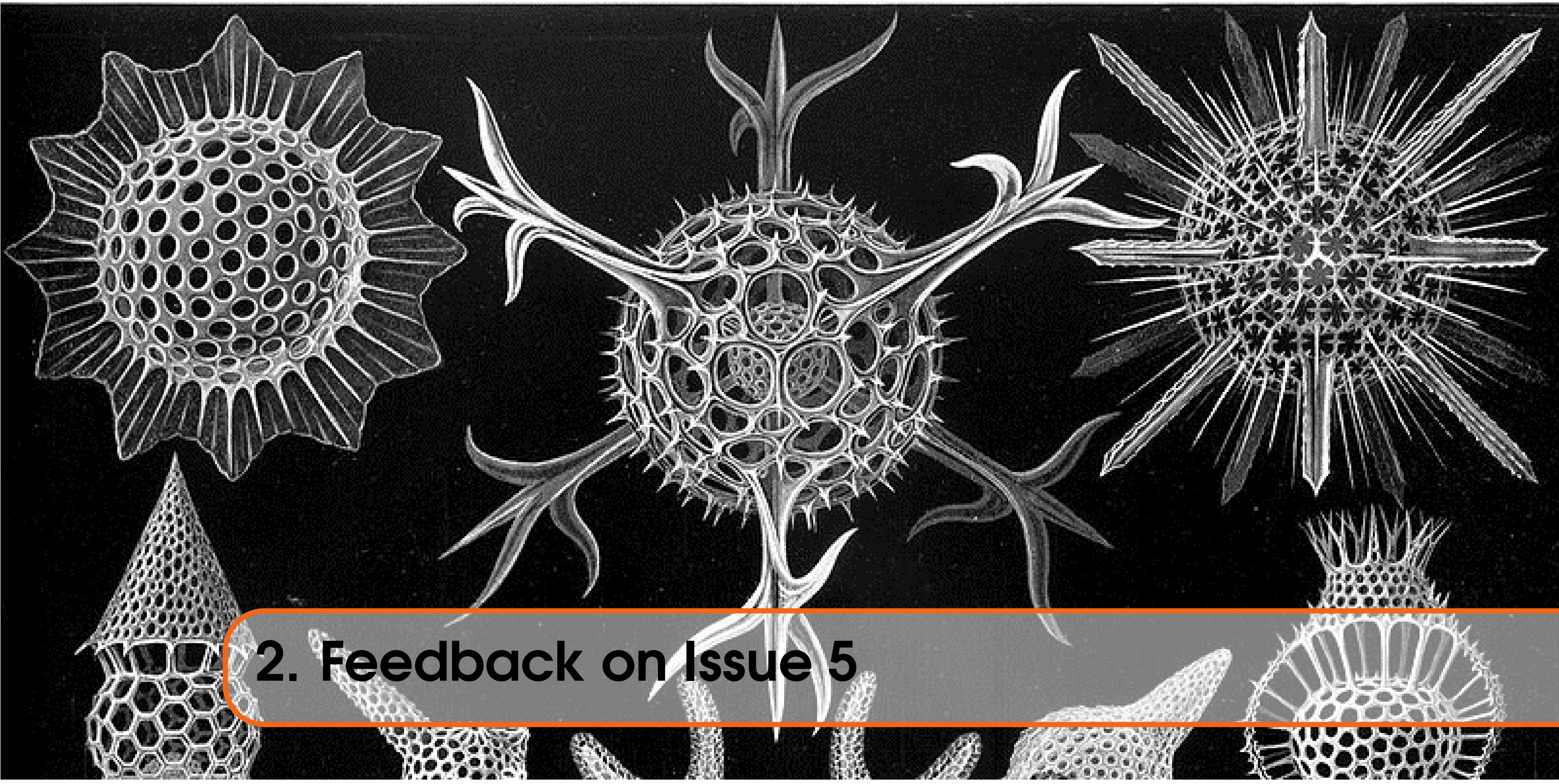
I'm rather hoping that this mailing list will not be a one-way affair, like QL Today appeared to be. I'm very open to suggestions, opinions, articles etc from my readers, otherwise how do I know what I'm doing is right or wrong?

I suspect George will continue to keep me correct on matters where I get stuff completely wrong, as before, and I know George did ask if the list would be contactable, so I've set up an email address for the list, so that you can make comments etc as you wish. The email address is:

`assembly@qdosmsq.dunbar-it.co.uk`

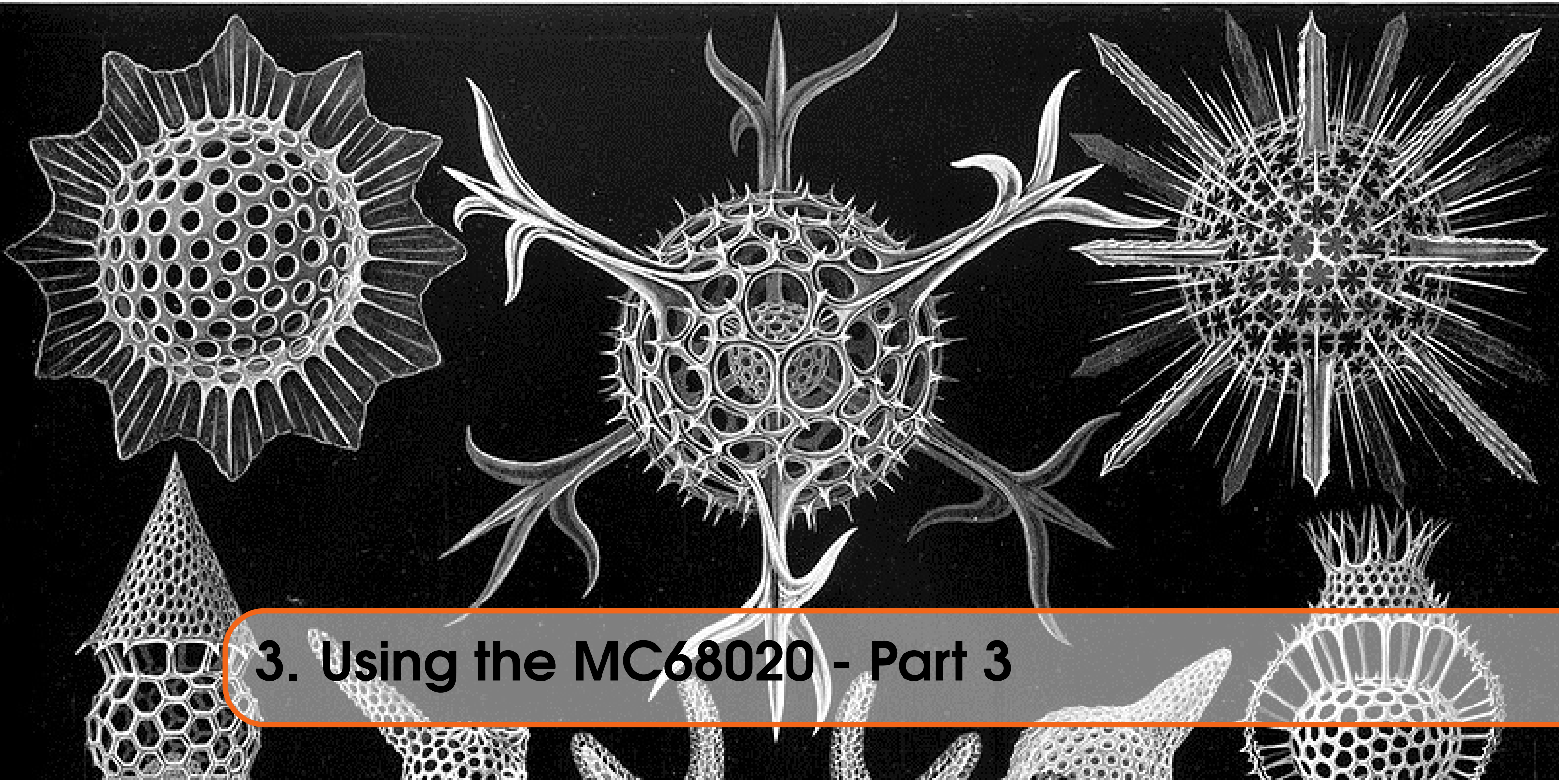
Any emails sent there will eventually find me. Please note, anything sent to that email address will be considered for publication, so I would appreciate your name at the very least if you intend to send something. If you do not wish your email to be considered for publication, please mark it clearly as such, thanks. I look forward to hearing from you all, from time to time.

If you do have an article to contribute, I'll happily accept it in almost any format - email, text, Word, Libre/Open Office odt, Quill, PC Quill, etc etc. Ideally, a LaTeXsource document is the best format, because I can simply include those directly, but I doubt I'll be getting many of those! But not to worry, if you have something, I'll hopefully manage to include it.



2. Feedback on Issue 5

2.1 No Feedback so far!



3. Using the MC68020 - Part 3

In the last issue, we took a very long look at the new and upgraded instructions that are now available when using an MC68020 processor as found in QPC - and possibly, in other emulators too. The old BBQL[1] uses an MC68008 and cannot cope with the new stuff.

To assemble these 62020 instructions, you need a copy of Gwass available from George's web site.[2]

This article continues our look at new features of the MC68020.

Here are the subjects I will cover in this issue, in relation to the 68020:

- The new format Status Register
- The various Control Registers used by the `MOVEC` instruction.
- New exception handlers.

3.1 Status Register

The status register looks like the following in the MV68020:

Bit															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
T_1	T_0	S	M	-	I_2	I_1	I_0	-	-	-	X	N	Z	V	C

Table 3.1: MC68020 Status Register

[1] Black Box QL

[2] http://gwiltprogs.info/page2.htm

3.1.1 Trace Bits T_1 and T_0

In the status register for the MC68020 we have now got an extra Trace bit - bit 14 - known as T_0. The original (MC68008) Trace bit, bit 15, is now known as the T_1 bit. Between the two Trace bits, better tracing can take place, as follows:

- 00 - When both Trace bits are zero, no tracing takes place.
- 01 - When T_1 is clear and T_0 is set, tracing takes place on a change of program flow - a branch, jump or subroutine call.
- 10 - When T_1 is set and T_0 is clear, tracing happens after every instruction. This is the tracing mode we are used to on the MC68008.
- 11 - Undefined. Probably best avoided!

3.1.2 Supervisor Master and Interrupt Modes

In addition to the extra Trace bit, there is a new Master bit as well. Bit 12 is the new Master bit.

On the MC68020, Supervisor mode is now split into two sub modes - master and interrupt. When the S and M bits are set then the processor is running in Master mode and uses the new Master Stack with the Master Stack Pointer in A7. (MSP(A7"))

When the S bit is set, and the M bit is clear, then the processor is running in Interrupt mode and uses another new stack, the Interrupt Stack, with A7 being the Interrupt Stack Pointer. (ISP(A7'))

The only difference between the two modes is the different stack pointer in use in register `A7`.

3.2 Control Registers and `MOVEC`

On the MC68020 we have the following control registers:

Control Register	Description
SFC	Source Function Code
DFC	Destination Function Code
USP	User Stack Pointer
VBR	Vector Base Register
CACR	Cache Control Register
CAAR	Cache Address Register
MSP	Master Stack Pointer
ISP	Interrupt Stack Pointer

Table 3.2: MC68020 Control Registers

3.2.1 SFC and DFC- Source and Destination Function Code

The alternate function code registers contain 3-bit function codes. Function codes can be considered extensions of the 32-bit logical address that optionally provides as many as eight 4-Gbyte address spaces - potentially increasing the 32 bit address bus to 35 bits.

The processor automatically generates function codes to select address spaces for data and programs at the user and supervisor modes.

Certain instructions use SFC and DFC to specify the function codes for operations.

The processor has three pins named FC0, FC1 and FC2. When the processor reads or writes from memory, these pins reflect information about the state of the processor.

They show the state of the processor - is it running in user or supervisor mode - and whether it is accessing data or instructions in memory.

The function codes are often used by external Memory Management Units (MMU) to protect various sections of memory. To the best of my knowledge, the QL doesn't have an MMU.

3.2.2 VBR - vector Base Register

The VBR is a 32 bit register which contains the base address of the exception vector table in memory. The displacement of an exception vector adds to the value in this register, which accesses the vector table.

On the MC68008, the exception table always lived at address 0, however, from the MC68010 onwards, the vector table still lives at address 0, but after a processor reset, the VBR can be adjusted to any desired location - provided that it can be addressed by a single 32 bit register.

3.2.3 CACR and CAAR - Cache Control

Many programs spend a lot of time executing loops. While within these loops, they execute the same (small) set of instructions over and over again. Each time the processor needs to execute an instruction, it must read it from memory.

There is a 256 byte instruction cache built in to the MC68020 (but probably not built in to the virtual MC68020 using in QPC, for example) which contains the most recently executed instructions.

In the case of a loop, the processor doesn't need to access memory to read the instructions more than once, in theory. When an instruction is read, it is stored in the cache and if executed again, will be read from cache which is much much quicker than reading from memory.

This is not always appropriate though, so the processor has the ability to enable, disable and otherwise manipulate the cache through the use of the CACR and CAAR control registers. These registers are 32 bits wide.

The use of these registers is beyond the scope of this series. They are unlikely to be mentioned ever again - except in passing, maybe!

3.2.4 USP, MSP and ISP - Stack Pointers

In normal user programs, the processor runs in user mode and the stack pointer in `A7` is the USP or User Stack Pointer.

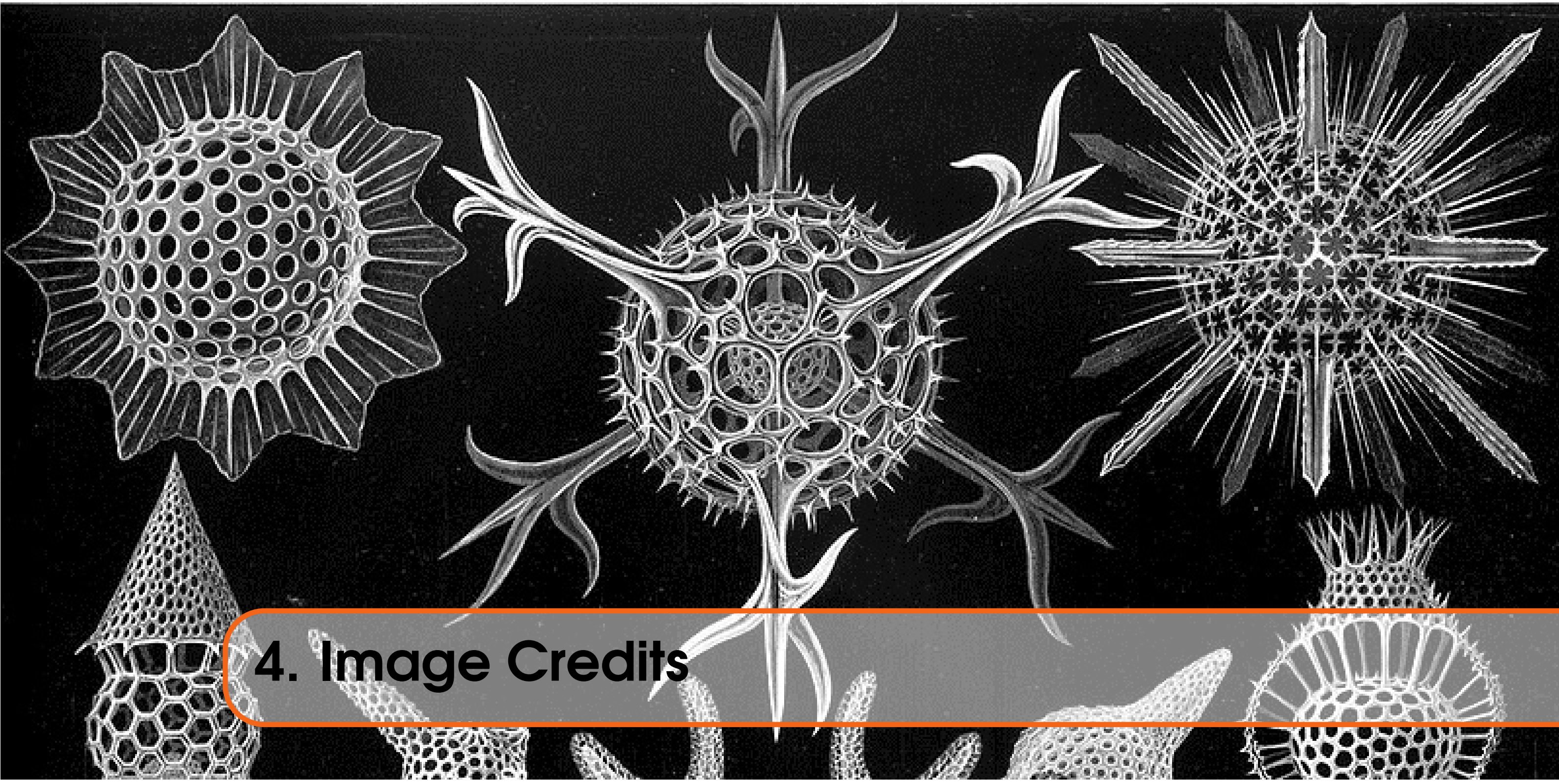
In Supervisor mode, a different stack is in use, usually limited in size, and on the BBQL, `A7` was then known as the SSP or Supervisor Stack Pointer.

On the MC68020 we have two submodes for Supervisor mode, and each one can have a different stack area and `A7` will be set accordingly to the Master Stack Pointer (MSP) or the Interrupt Stack Pointer (ISP) depending on the settings of the S and M bits in the Status Register.

If S is set and M is clear, the ISP is in `A7`, while the MSP is in `A7` if both bits are set.

3.3 Exception Handlers

???????????????????????



4. Image Credits

The front cover image on this ePeriodical is taken from the book *Kunstformen der Natur* by German biologist Ernst Haeckel. The book was published between 1899 and 1904. The image used is of various *Polycystines* which are a specific kind of micro-fossil.

I have also cropped the image for use on each chapter heading page.

You can read about Polycystines on Wikipedia and there is a brief overview of the above book, also on Wikipedia, which shows a number of other images taken from the book. (Some of which I considered before choosing the current one!)

Polycystines have absolutely nothing to do with the QL or computing in general - in fact, I suspect they died out before electricity was invented - but I liked the image, and decided that it would make a good cover for the book and a decent enough chapter heading image too.

Not that I am suggesting, *in any way whatsoever*, that we QL fans are ancient.