Great Australian software developers: Nick Gammon & G-Pascal

web.archive.org/web/20180706003949/http://www.supercoders.com.au:80/blog/nickgammongpascal.shtml

- **Mon 27 Jun 2011**

Australian software developers include some of the worlds most talented. This blog post celebrates the work of Nick Gammon, an Australian software developer who in 1983 wrote a pascal compiler for the Commodore 64 and Apple 2 computers. It ran in 16K of RAM. Here's how he did it.

Imagine you are a software developer being briefed on your next project:

You:	Hi boss, I believe you have a new project for me?
Boss:	Yes, please sit down. You're our star programmer so I have decided to give this exciting new project to you. I want you to write a compiler.
You:	A compiler, boss?
Boss:	Yes, a Pascal compiler.
You:	Oh. A Pascal compiler. I see. Err….what language do you want me to write it in?
Boss:	This one needs to be fast. I want you to write it in assembly language.
You:	Assembly language, boss?
Boss:	Yes assembly language.
You:	You want me to write a Pascal compiler in assembly language?
Boss:	Yes
You:	*(pauses)*…..How much memory does it need to fit in boss?
Boss:	16K
You:	*(coughs, shifts nervously)* 16K? A Pascal compiler in 16 kilobytes boss?
Boss:	Kilobytes, yes. 16 kilobytes.
You:	*(slow pause)* Boss, 16K isn't much memory. The logo image on our website is 64 kilobytes and you want a complete Pascal compiler in one quarter that size?
Boss:	Yes indeed.
You:	Anything else I should do while I'm at it?
Boss:	Well, it really does need an integrated editor – can you build that too and fit it in the 16K? And documentation – you'll need to write a user manual. I'll need you to write sample programs and do all the advertising and marketing to sell it. I'd like it to run on two hardware platforms too.
You:	Umm, that seems rather ambitious.

Boss:	How long will it take you to get it all finished?
You:	*(looks sideways, darts for the door, runs screaming from room!)*

Seems like quite a challenge, one that would probably send most modern programmers screaming from the room. How many software developers do you know who could write a compiler in assembly language with an integrated editor and make it fit in 16 kilobytes of RAM? How many could then write a comprehensive user manual, sample programs and take it to market doing advertising and writing magazine articles explaining how it works?

I'm guessing not many.

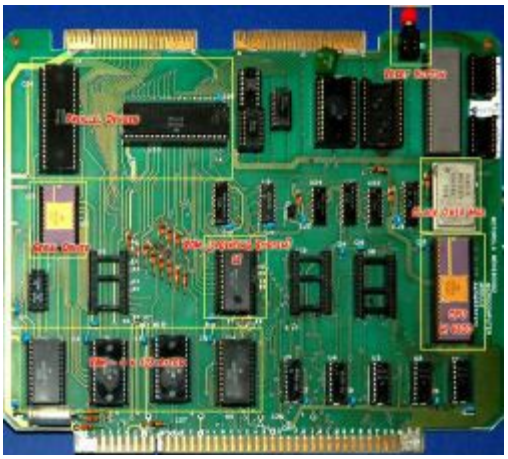
But that's exactly what Australian software developer Nick Gammon did when he wrote 'G-Pascal' for the 6502 based Apple II and Commodore 64 home computers. He did it back in 1983.

It's a magnificent feat of software development and I recently asked Nick to tell me the story behind the development of G-Pascal. Here is Nick's tale in his own words:

In the late 1970's I was using a hand-soldered computer based on a Motorola 6800 Evaluation Kit. This predated pre-assembled things like the Tandy TRS-80 or the Commodore Pet. Basically it was well before personal computers were commonly heard of in Australia. Here is a photo of the old Motorola board:

This 'PC' had about 16Kb of RAM (from memory) and 16 Kb of EPROM. Here is the board with the EPROM's on it:

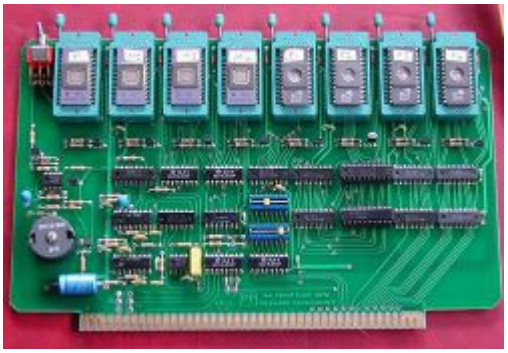
The extra 16 Kb RAM board & another 16K "Mother Board":



My first task was to write an assembler and editor, as it had virtually no software with it, except for a BIOS in ROM - about 256 bytes or so.

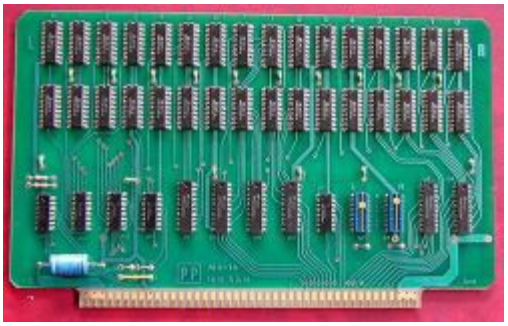
Since it didn't have an assembler to start with, I had to write the assembler in hex machine codes initially. Once that worked, each version of the assembler could be used to develop the next one. The editor/assembler took up 8 KB between them, so each version used up half of the onboard EPROM, and I used that half to burn the next version in the other half, and then swap them over to "upgrade" it. Here's how I programmed the assembler in hex:

Although I had written an assembler for my Motorola PC in October 1978, I had no experience in writing compilers, and was becoming increasingly interested in adding a proper high-level language to its repertoire.



I had been using BPL (Burroughs Programming Language) at the place where I worked, and found the structured approach in such a language much easier to work with than assembler code.

I had no idea, however, how to go about writing such a thing. That was when I spotted an article about 'A Tiny Pascal Compiler', an article written by Kin-man Chung and Herbet Yuen, published in the September, October and November 1978 issues of Byte Magazine (refer to links at bottom of this page). They described how they had made a cut-down version of Pascal, implemented in the Basic language. It sounded interesting but was apparently very slow, no doubt as a result of trying to write a compiler in Basic. They reported that their compiler compiled about 8 lines per minute.



The useful thing about this article was that they described in great detail about how such a compiler would be implemented, gave syntax diagrams, and had the source code of their compiler. Having read the article I thought I understood how it worked, and decided to do my own version, in assembler, on my little PC. It was quite a long project, but at least I already had a text editor, so I could get straight on with the compiler.

The compiler actually generated P-Codes (Pseudo-machine code) which was not real machine code but code for a hypothetical machine, which an 'interpreter' then processed. This simplified things somewhat as I did not need to be concerned with actually generating machine instructions.

Thus, the running a program consisted of two steps: first, compiling the source code into P-Codes, and secondly, interpreting the machine codes. The compile step only had to be done once, so to repeatedly run a program you could go straight to the interpret step each time.

After a few weeks' work, I had it working, and it turned out to be much faster than their 8 lines per minute. Mine worked at around 2,000 lines per minute, which meant that even a largish program (eg. 200 lines) would compile in about 6 seconds. This I also put into EPROM (it took 8 Kb) so that my 16 Kb EPROM board was now full. It had:

*1 Editor: 4 Kb
*2 Assembler: 4 Kb
*3 Pascal Compiler and Interpreter: 8 Kb

This was the first, (and only that I know of), computer that had a Pascal compiler in ROM (EPROM), so that after turning it on you could program in Pascal instantly.

Unfortunately, my editor, assembler and compiler were designed rather specifically around my exact configuration (keyboard, monitor card, memory configuration), and thus I was not in a position to sell it. Thus, it was just a once off.

Saving of files (like the source code) was to a magnetic tape - I had a tape-recorder connected to the PC, and had to manually hit "record" and then start the save process.

In 1982 I thought it was time to try to make some money out of my ability to write compilers, so I decided to 'port' my Pascal compiler from the Motorola 6800 to the Apple II. This was not a totally trivial task, as the two computers had different chips in them. The Motorola had a 6800 chip, and the Apple had a 6502. The instructions were different but not completely different. For example, the 6502 did not have a B register.

After a few weeks' work I had the Pascal compiler working, so I decided to call it G-Pascal. The 'G' stood for Gammon. I once tried to register G-Pascal as a trademark, but apparently the letter 'G' was not enough of a distinguishing mark. In other words, you can't trademark individual letters of the alphabet. Interestingly I believe Apple these days are able to trademark the iPhone, which is basically just the letter "i" attached to an existing word "Phone". Since they can protect that mark, but I could not trademark "G" attached to "Pascal" perhaps they had better lawyers than me.

The program only took up 12 Kb of memory when loaded. In that space was the editor, compiler and P-code interpreter. It didn't need the assembler, as that was only needed to compile the G-Pascal.

I wrote a user manual for G-Pascal, and printed it out on our dot-matrix (Brother Microline 84) printer. We then photocopied a few dozen copies, and used a binding machine to put ring binders on them. We bought some plastic bags from Venus Hartung, copied some disks, and were ready to sell!

I took out ads in the Australian Personal Computer magazine (refer to links), and also Your Computer (refer to links), I think. I am as good an advertising copywriter as a salesman, so my ads were not very enticing (refer to links). I concentrated on giving as much technical detail, in fine print, as I could, whereas I think a better approach would be to make a more eye-catching ad, and mail details to people when they expressed interest.

Still, a few dozen copies of G-Pascal were sold. They may have covered the cost of the advertising, but I doubt it.

In August 1982, the Australian-based 'Your Computer' magazine printed a program I wrote to demonstrate writing an adventure game in G-Pascal (refer to links). The article ran to 6 pages of fine print, taking up quite a bit of room in the magazine and giving us lots of exposure.

We also got a favourable review of G-Pascal (for the Apple II) in August 1982. It was described as an 'interesting and extremely useable small language package', 'compact but particularly potent package', and 'amazing value for money'.

In April 1983 I was contacted by Ian Webster, the editor of Australian Personal Computer magazine, and met him at some trade show or other.

He had seen my G-Pascal for the Apple II, and suggested that the new state-of-the-art computer would be the Commodore 64. As it had the same chip as the Apple II (the 6502) he suggested I port G-Pascal to the Commodore 64.

I bought a Commodore 64, and I also developed a method of transferring files from the Apple II to the Commodore 64 via the parallel port. This meant I could keep using the assembler that I had purchased for the Apple, but assemble programs for running in the Commodore 64. At this stage my Apple II had a floppy disk drive, and the assembler was more potent (Merlin I think it was called), which simplified the assembly part of the process.

The transfer method involved loading a small 'transfer' program onto both PCs, and then initiating a 'send' or 'receive'. Transferring via the parallel port was very fast, so the whole G-Pascal could be sent in a couple of seconds.

The Commodore 64 version had extra features to support the hardware on that PC. In particular, I added stuff to let you handle sprites, the colour screen, and the sound chip. This extra stuff took about another 4 Kb, so I think the whole G-Pascal took up 16 Kb.

In the November 1983 issue of "Your Computer" we got a very favourable review of G-Pascal for the Commodore 64 (refer to links). Amongst other things the reviewer said our program 'provides outstanding support for all the features of the system', 'support for graphics, sprites and sound is superb' and finally 'you may have heard that Australian programmers are as good as any, here is the evidence'.

In the end, I think I sold around 2000 copies of G-Pascal. It made some money certainly, but quite a bit was swallowed up by advertising, and I probably would have made more by just working full-time, so I eventually went back to doing that.

After working full-time for a long time, I got interested in MUD games, and wrote my own MUD client (MUSHclient) which I presume you have seen on my web site. That is still my ongoing interest.

More recently I discovered the Lua scripting language, which is now incorporated into MUSHclient.

The funny thing is, as soon as I saw the Lua syntax I thought "oho! this looks like Pascal". The syntax is certainly remarkably similar. The repeat ... until loops, the while ... end loops, etc. There are differences, certainly, but somehow seeing Lua reminds me of my G-Pascal days. :)

Lua also uses the P-code approach - it compiles into P-codes and interprets them, like G-Pascal did.

Unfortunately I think I threw out the old hand-written assembler. That was an interesting job - coding an assembler using pencil and paper. I got good at memorizing the hex codes for common instructions.

That’s Nick’s tale of G-Pascal. It’s a wonderful story of the 'good old days' and a testament to his

programming talent.

If you're interested you can try out G-Pascal yourself by downloading the G-Pascal Commodore 64 disk image To use it, you'll need a Commodore 64 emulator such as VICE which is free and open source. The full G-Pascal user manual is below.

Some exciting news - Nick has dug through his archives and found the assembly language source code for G-Pascal for the Commodore 64, he has open sourced it and you can download it here.

These days you can find Nick on the forums at his website where he continues work on his MUD and MUSH game software. He's also doing interesting stuff with Arduino such as this 'Adventure Game On A Chip' If you're doing Arduino development you might run into him at the Arduino forum

Here's to Nick Gammon, a great Australian software developer.

Do you know an Australian software developer who has achieved something remarkable? If yes, please let me know at andrew.stuart@supercoders.com.au as I'm looking to celebrate the work of more great Australian software developers.

Links- caution some of these files are very large:

G-Pascal user manual for Commodore 64 9.4M PDF
G-Pascal advertisement 1 6.3M PDF
G-Pascal advertisement 2 92K PDF
G-Pascal advertisement 3 136K PDF
G-Pascal advertisement 4 107K PDF
G-Pascal advertisement 5 111K PDF
G-Pascal Adventure Your Computer 1.8M PDF
G-Pascal Apple Brochure 1 356K PDF
G-Pascal Apple Brochure 2 555K PDF
G-Pascal Apple manual cover 98K PDF
G-Pascal Apple Manual 3.7M PDF
G-Pascal article Commodore Magazine 446K PDF
G-Pascal communications program 1.8M PDF
G-Pascal Disk 1.9M PDF
G-Pascal manual Commodore 64 front cover 1.4M PDF
G-Pascal newsletter 1 1.3M PDF
G-Pascal newsletter 2 2.8M PDF
G-Pascal newsletter 3 4.2M PDF
G-Pascal newsletter 4 4.4M PDF
G-Pascal newsletter 5 4.4M PDF
G-Pascal review Australian Microcomputer Magazine 732K PDF
G-Pascal review Your Computer 958K PDF
G-Pascal review Bits & Bytes 1.4M PDF

andrew.stuart@supercoders.com.au

Thanks Nick working on this together was fun!

Apple 2 image sourced from http://oldcomputers.net/appleii.html all rights reserved
Commodore 64 image sourced from http://http://www.forevergeek.com/ all rights reserved

Copyright © SuperCoders 2010 - 2011 All Rights Reserved.