The Macroassembler AS

See here for a Danish translation of this text provided by Daniela Milton.

How it all begun..

Surely some of you already thought about writing own programming tools - maybe just a small converter, or a full blown environment because the tools you currently use simply suck. Such a project may look gorgeous and impractible for a single person with limited programming experience, but it's possible - do it step by step and don't expect that things develop the way you thought in the beginning. Here's the story how AS came to existence, a sequence of coincidences and false anticipations:

A few years ago, the german computer magazine 'mc' (which no longer exists) published an article about a project called *PcPar68000*. The idea was to build a parallel computer from cheap, easily obtainable components. The individual CPU boards consisted of an MC68000 CPU, 64 Kbytes of RAM and a bunch of TTL logic. No PALs, no ASICs, just standard components. The individual CPUs didn't have a ROM - their RAM was mapped into the PC's address space so the PC could write the program into the 68000's RAM and then take it out of reset state. Each CPU was on a separate board, so the parallelism was limited by the number of free ISA slots you had in your PC.

The total price of the components was about DEM100, mainly driven by the processor and the two 32Kx8 SRAMs - at that time, they made up about two thirds of the total costs! Since my first intend was only to experiment with the 68K architecture, I even saved the cost for the PCB by wrapping my first CPU board. The only thing I really bought from mc was the assembler, a simple non-relocatable assembler without macros. It costed me about DEM50, and I thought this to be a fair price...

...until I started working with the thing! The assembler had some severe bugs (e.g. not detecting certain erroneous constructs, it simply generated junk) that made working with it more and more difficult, so at some point I decided to look how difficult writing an own one would be.

The initial target was just to write an assembler that was compatible with the one from mc. I started writing a formula parser, which seemed to me to be the hardest part. Again a few years ago, I already had written an 8048 assembler on my Apple][running under CP/M. That one lacked a formula parser, i.e. arguments to instructions were limited to either simple constants or symbols. Once the parser was up and tested, the next step was the design of the symbol table. I chose a binary tree as the basic data structure, since I was already quite familiar with pointers and dynamic data structures, and you didn't have any obscure 64K limits that would have resulted from using something like a table (don't forget that at that time, the PC user's choice of operating systems was limited to MS-DOS - and my fastest PC at that time was a PC/XT with 8 MHz. There were commercial Unices from several vendors, but only at horrendous prices).

Once the basics were laid (engineers tend to choose a bottom-up approach...), I wrote a line parser, a few lines of code to output the results in the proper file format (fortunately, the original assembler's output format was documented), and started to march through the 68000's instruction set. The parser for the 68K's address expressions proved to be a bit difficult, but otherwise it was just a question of diligence. Finding the correct mnemonic was just a giant ladder of `IF OpPart='...' THEN`, which may seem stupid and brute force at the first look, but it can be quite effective if you put the most frequent instructions *first*. This technique is still used in a lot of AS's code generators, and I just started a few months ago to remove this in favour of hash tables.

The result was a simple 68000 cross assembler that did everything the original assembler did, just without the annoying bugs :-) I was happy, and the first milestone was done. So what next?

The first extension was to extend the supported instruction set. At that time, I had plans to build new CPU boards with a 68020 and an FPU (my main application on the PcPar68000 was to calculate Mandelbrot sets ;-)), but this never came to reality due to the price of the CPU/FPU at that time. The extensions to my assembler however didn't cost me anything, and I did them in anticipation that I would need them someday.

Of course, the assembler needed a feature to block me from accidentally using 68020 instructions on a 68000, and that's how the `CPU` statement saw the light!

The other extension I made to the assembler were macro instructions. It proved to be tricky to feed the expanded lines from the macro processor back to the parser, and early users of AS might remember that AS was quite picky about nested macro statements...

This was the first chapter of the AS story, and AS (which was named ASM68 at that time) stayed like this for two years. It did everything I needed, and I didn't dare to hand it out to others. In fact, the only other person that ever got a copy of ASM68 was Bernhard Zschocke, who needed something to reuse his DTACK-GROUNDED board after he switched from an Apple][to a PC.

Coincidentally, it's the same person who triggered the process that made AS out of ASM68. Somewhere in 1991/1992, it was time for me as a student of electrical engineering at RWTH Aachen to take one of the practical courses (you need them on the way to a diploma), and I chose a course in microprocessor control. It was headed by Bernhard Zschocke, and the course's target was to control a robot arm with a CP/M system. My results in this course were quite good, and Bernhard and I became friends over time.

In the fall of 1992, it was finally time to throw out the ancient CP/M boxes (when was the last time you've seen an 8 inch floppy?) and to replace the environment with something more en-vogue. The development platform were now ATs, and the target was a 80535-based microcontroller board. Of course, you also needed an assembler for this beast. No problem? Well... On the Z80 machines, we had used the M80 from Microsoft. This was a fully commercial assembler, with the result that students who wanted to do their exercises at home either had to buy one or work with an (illegal) copy. This should change with the new platform! Bernhard knew Intel's assembler from other projects, but this was of course also commercial, and Web-based searches weren't available at that time. There was another student colleague of Bernhard (don't know his name) who sold a selfmade 8051 assembler, but he refused to release a special shareware versions for the students. This reminded Bernhard that he once got an assembler from me, so "Can't you rewrite it to generate 8051 code?" Ahem...

One of my largest problems is that I can't say 'No', so I found myself at the end of 1992 with the task of modifying ASM68 for the 8051 target platform. I quickly realized that it would basically suffice to exchange the 68000 parser with something that understands 8051 code. But why throw it out when I already can switch between different 68K targets? Just add another branch! Over Christmas 1992, the first version 1.30 of AS (I continued to use the version numbering from ASM68) was made, with 68xxx, 6502, 8048, and 8051 as target platforms. Really new was the User's Manual (I hadn't bothered witing a manual for ASM68, since I was the only user) and a set of utilities to convert the assembler's output to binary and HEX files. The Z80 followed shortly later. The students at the course were my first beta testers, and they were a source for bug reports and ideas for years until Bernhard left RWTH Aachen.

Since AS is free, the students not only took it home to do their exercises, they also gave it to their friend, which widened the 'user community'. Of course, they had different uses for AS and brought in other ideas. That's how AS began to grow...and it grows until today!

And what about the PcPar68000? Well, the 19 inch rack with the CPU boards still has a place of honour on my desk at home, but I haven't turned it on for years...I would probably need a few days to get it going again. I collected a bit of information and photos on a separate page which is here.
