README

MMIX is a computer intended to illustrate machine-level aspects of programming. In my books *The Art of Computer Programming*, it replaces MIX, the 1960s-style machine that formerly played such a role. MMIX's so-called RISC ("Reduced Instruction Set Computer") architecture is much better able to represent the computers being built at the turn of the millennium.

I strove to design MMIX so that its machine language would be simple, elegant, and easy to learn. At the same time I was careful to include all of the complexities needed to achieve high performance in practice, so that MMIX could in principle be built and even perhaps be competitive with some of the fastest general-purpose computers in the marketplace. I hope that MMIX will therefore prove to be a useful vehicle for people who are studying how to improve compilers and operating systems, and that other authors will like MMIX well enough to make use of it in their own textbooks. My goal in this work is to provide a clean, complete, and well-documented "machine-independent machine" that people all over the world will be able to use as a testbed for long-term research projects of lasting value, even as real computers continue to change rapidly.

This book is a collection of programs that make MMIX a virtual reality. One of the programs is an assembler, MMIXAL, which converts MMIX symbolic files to MMIX object files. There also are two simulators, which execute the programs in given object files. The first simulator, called MMIX-SIM or simply MMIX, executes a program one instruction at a time and allows convenient debugging. The second simulator, MMMIX, simulates a high-performance pipeline in which many aspects of the computation are overlapped in time. MMMIX is in fact a highly configurable "meta-simulator," capable of simulating an enormous variety of different kinds of pipelines with any number of functional units and with many possible strategies for caching, virtual address translation, branch prediction, super-scalar instruction issue, etc., etc.

The programs in this book are somewhat primitive, because they all are based on a simple terminal interface: Users type commands and the computer types out a reply. Still, these programs are adequate to provide a basis for future developments. I'm hoping that at least one reader of this book will discover how much fun MMIX programming can be and will be motivated to create a nice graphical interface, so that other people will more easily be able to join in the fun. I don't have the time or talent to construct a good GUI myself, but I've tried to write the programs in such a way that modifications and enhancements will be easy to make.

The latest versions of all these programs can be downloaded from MMIX's home page

`http://www-cs-faculty.stanford.edu/~knuth/mmix-news.html`

in a file called `mmix.tar.gz`. The programs are copyrighted, but anyone can use them without charge. Furthermore I explicitly allow anybody to copy and modify the programs in any way they like, provided only that the computer files are given different names whenever they have been changed. Nobody but me is allowed to make a correction or addition to the copyrighted file `mmixal.w`, for example, unless the corrected file is identified by some other name (possibly '`turbo-mmixal.w`' or '`mmixal++.w`', etc.).

The programs are all written in `CWEB`, a language that combines C with $\TeX$ in such a way that standard preprocessors can easily convert `mmixal.w` into a compilable file `mmixal.c` or a documentation file `mmixal.tex`. `CWEB` also includes a "change file" mechanism by which people can easily customize a master source file like `mmixal.w` without changing the master file in any way. (See

`http://www-cs-faculty.stanford.edu/~knuth/cweb.html`

for complete information about `CWEB`, including installation instructions for the related software.) Readers of the present book who are unfamiliar with `CWEB` might want to refer to the notes on "How to read `CWEB` programs" that appear on pages 70–73 of my book *The Stanford GraphBase* (New York: ACM Press, 1993), but the general ideas are almost self-explanatory so I decided not to reprint those notes here.

During the next several years, as I write Volume 4 of *The Art of Computer Programming*, I plan to prepare updates to Volumes 1–3 whenever Volume 4 needs to refer to new material that belongs more properly in earlier volumes. These updates, called "fascicles," will be available on the Internet via

`http://www-cs-faculty.stanford.edu/~knuth/taocp.html`

and they will also be published in hardcopy form. The first such fascicle is already finished and available for downloading; it is a tutorial introduction to `MMIX` and the `MMIX` assembly language. Everybody who is seriously interested in `MMIX` should read that First Fascicle, preferably before reading the programs in the present book.

I've tried to make the `MMIX`ware programs interesting to read as well as useful. Indeed, the MMIX-PIPE program, which is the chief component of the MMMIX meta-simulator, is one of the most instructive programs I've ever had the pleasure of writing. But I don't expect a great number of people to study every part of this book closely, or even to study every part of MMIX-PIPE. The main purpose of this book is to provide a complete documentation of the `MMIX` computer and its assembly language. Many details about `MMIX` were too "picky" or too system-oriented to be appropriate for the First Fascicle, but every detail about `MMIX` can be found in the present book.

After the `MMIX`ware programs have been installed on a UNIX-like system, they are typically used as follows. First a user program is written in assembly language and put into a file, say `foo.mms`. (The suffix `.mms` stands for "`MMIX` symbolic.") Then the command

```
  mmixal foo.mms
```

will translate it into an object file, `foo.mmo`. Alternatively, a command such as

```
  mmixal -l foo.lst foo.mms
```

could be used; this would produce a listing file, `foo.lst`, in addition to `foo.mmo`. The listing file, when printed, would show the contents of `foo.mms` together with the assembled machine language instructions.

Once an object file like `foo.mmo` exists, it can be run on the simple simulator by issuing a command such as

```
mmix foo
```

(or `mmix foo.mmo`). Many options are also possible; for example,

```
mmix -s foo
```

will print running time statistics when the program ends;

```
mmix -P foo
```

will print a profile that shows exactly how often each instruction was executed;

```
mmix -v foo
```

will give "verbose" details about everything the simulator did;

```
mmix -t2 foo
```

will trace each instruction the first two times it is performed; etc. Also

```
mmix -i foo
```

will run the simulator in interactive mode, obeying various online commands by which the user can watch exactly what is happening when key parts of the program are reached. The command

```
mmix foo bar
```

will run the simulator as if `MMIX` itself were running the command '`foo bar`' with a rudimentary operating system; any number of command-line arguments can follow the name of the program being simulated.

The MMMIX meta-simulator can also be applied to the same program, although a bit more preparation is necessary. First the command

```
mmix -Dfoo.mmb foo bar
```

will dump out a binary file `foo.mmb` containing the information needed to load '`foo bar`' into `MMIX`'s memory. Then a command like

```
mmmix plain.mmconfig foo.mmb
```

will invoke the meta-simulator with a "plain" pipeline configuration. The meta-simulator always runs interactively, using the prompt '`mmmix>`' when it wants instructions about what to do next. Users can type '`?`' in response to this prompt if they want to be reminded about what the simulator can do. Typical responses are '`vff`' (run verbosely); '`v0`' (run quietly); '`p`' (show the pipeline); '`g255`' (show global register 255); '`D`' (show the D-cache); '`b200`' (pause when location #`200` is fetched); '`1000`' (run 1000 cycles); etc. Some familiarity with MMIX-PIPE is necessary to understand the meta-simulator's reports of its activity, but users of `mmmix` are assumed to be able to extract high-level information from a mass of low-level details. (This talent, after all, is the hallmark of a computer scientist.)

The programs in this book appear in alphabetical order:

MMIX explains everything about the `MMIX` architecture.

MMIX-ARITH contains subroutines for 64-bit fixed and floating point arithmetic, using only 32-bit fixed point arithmetic.

MMIX-CONFIG processes configuration files for MMMIX.

MMIX-IO contains subroutines for the primitive input/output operations of a rudimentary operating system.

MMIX-MEM handles memory references of MMMIX in special cases associated with memory-mapped input/output.

MMIX-PIPE does the hard work of pipeline simulation.

MMIX-SIM is the program for the non-pipelined simulator.

MMIXAL is the assembly program.

MMMIX is the driver program for the meta-simulator.

MMOTYPE is a utility program that translates an `MMIX` object file into human-readable form.

The first of these, MMIX, is not actually a program, although it has been formatted as a `CWEB` document; it is a complete definition of `MMIX`, including the details of features that are used only by the operating system. It should be read first, but the other programs can be read in any order. (Actually MMIXAL or MMIX-SIM should probably be read next after MMIX, and MMIX-PIPE last. The program MMIX-SIM is the line-at-a-time simulator that is known simply as `mmix` after it has been compiled.)

Mini-indexes have been provided on each right-hand page of this book so that the programs can be read essentially as hypertext. Every identifier that is used on a two-page spread but defined on some other page is listed in the mini-index. For example, a mini-index entry such as '*oplus*: **octa** ( ), MMIX-ARITH §5' means that the identifier *oplus* denotes a function defined in section §5 of the MMIX-ARITH module, returning a value of type **octa**. A master index to all uses of all identifiers appears at the end of this book.

I've tried to make this book error-free, but I must have blundered at least once. Therefore I shall use part of Internet page

`http://www-cs-faculty.stanford.edu/~knuth/mmixware.html`

as a bulletin board for posting all errors that are known to me. The first person who finds a hitherto unreported error will be entitled, as usual, to a reward of $2.56, gratefully paid.

Happy hacking!

Donald E. Knuth
Cambridge, Massachusetts
17 October 1999