Havm

The Tree Virtual Machine
13 May 2014, HAVM Version 0.26a

Robert Anisko

Table of Contents

HAVM is a virtual machine designed to execute simple register based high level intermediate code. It is based on the intermediate representations ("canonicalized" or not) defined by Andrew Appel in his "Modern Compiler Implementation". It is nevertheless generic enough so that any (student) compiler could target its intermediate language to HAVM's language.

Its features are:

- two object types, integers and pointers
- tree-like source language (two way conditional jumps, arbitrarily nested subroutines calls, etc.)
- threaded source language (one way conditional jumps, etc.)
- a runtime library comparable to SPIM's
- a debugging mode displaying the instructions being executed

It was written by Robert Anisko as an LRDE member, so that EPITA students could exercise their compiler projects before the final jump to assembly code. It is implemented in Haskell, a pure non strict functional language very well suited for this kind of symbolic processing. HAVM was coined on both Haskell, and vm standing for Virtual Machine.

Information about HAVM can be found on HAVM Home Page[1], and feedback can be sent to lrde's Projects Address[2]. LRDE stands for Laboratoire de Recherche et Dévelopment de l'EPITA[3], i.e., the Research and Development Lab of EPITA, the École Pour l'Informatique et les Techniques Avancées[4].

Andrew Appel's home page[5] includes links to material related to compilers theory, and some information about the Modern Compiler Implementation[6] book series.

More information on Haskell can be found on Haskell Home Page[7]. HAVM requires a specific Haskell compiler, GHC, the Glasgow Haskell Compiler[8].

---

[1] HAVM Home Page, `http://www.lrde.epita.fr/wiki/Havm`.

[2] lrde's Projects Address, `projects@lrde.epita.fr`.

[3] Laboratoire de Recherche et Dévelopment de l'EPITA, `http://www.lrde.epita.fr`.

[4] École Pour l'Informatique et les Techniques Avancées, `http://www.epita.fr`.

[5] Andrew Appel's home page, `http://www.cs.princeton.edu/~appel/`.

[6] Modern Compiler Implementation, `http://www.cs.princeton.edu/~appel/modern/`.

[7] Haskell Home Page, `http://www.haskell.org`.

[8] Glasgow Haskell Compiler, `http://www.haskell.org/ghc/`.

1 Invoking HAVM

To invoke `havm` run

```
havm options file
```

where '`file`' is a simple text file, and *options* is any combination of the following options:

'`-h`'
'`--help`' Display a help message and exit successfully.

'`-V`'
'`--version`'
Display the version number and exit successfully.

'`-d`'
'`--display`'
Unparse the content of the *file* on the file descriptor *fd*, defaulting to 2 (standard error output).

'`-p [fd]`'
'`--profile[=fd]`'
Report simple profiling information on *fd*, defaulting to 2 (stderr).

'`-t [fd]`'
'`--trace[=fd]`'
Display each instruction before execting it on *fd*, defaulting to 2 (stderr).

'`-l [fd]`'

'`--low[=fd]`'
Reject high level constructs. See Section 2.4 [LIR], page 8, for a description of the valid subset.

2 The HAVM Language

HAVM supports two different source languages, HIR and LIR, the second being a subset of the first one.

2.1 HIR

In both languages, white spaces are ignored, and comments are introduced by `#` and end at the end of line, or opened by `/*` and closed by the next `*/`. HIR is defined by the following grammar:

```
Exp ::= "const" int
     |  "name" Label
     |  "temp" Temp
     |  "binop" Oper Exp Exp
     |  "mem" Exp
     |  "call" Exp [{Exp}] "call end"
     |  "eseq" Stm Exp
     .

Stm ::= "move" Exp Exp
     |  "sxp" Exp
     |  "jump" Exp Label
     |  "cjump" Relop Exp Exp Label Label
     |  "seq" [{Stm}] "seq end"
     |  "label" Label
     |  "label" Label Literal
     .

Oper  ::= "add" | "sub" | "mul" | "div"
       |  "and" | "or" | "lshift" | "rshift" | "arshift" | "xor"
Relop ::= "eq" | "ne" | "lt" | "gt" | "le" | "ge"
       |  "ult" | "ule" | "ugt" | "uge"
Label ::= Ident
Temp  ::= fp | rv | sp | Ident
fp    ::= "fp" | "$fp"
sp    ::= "sp" | "$sp"
rv    ::= "rv" | "$v0"
Ident ::= [$a-zA-Z_][$a-zA-Z_0-9]*
```

In addition, the following alternative syntax for operators is supported, but deprecated.

```
Oper  ::= "(+)" | "(-)" | "(*)" | "(/)"
Relop ::= "(=)" | "(<>)" | "(<)" | "(>)" | "(<=)" | "(>=)"
```

A `Literal` is almost a Tiger string: it is enclosed by ‘`"`’, with support for the following escapes:

‘`\a`’, ‘`\b`’, ‘`\f`’, ‘`\n`’, ‘`\r`’, ‘`\t`’, ‘`\v`’
    control characters.

\x*num* The character which code is *num* in hexadecimal (upper case or lower case or mixed). *num* is composed of exactly 2 hexadecimal characters.

'\\' A single backslash.

'\'' A simple quote.

'\"' A double quote.

character
If no rule above applies, this is an error.

In addition the following restriction must be respected:

`call` A `call` has always the form '`move (name l, ...)`'.

`cjump` Destinations must be valid: '`cjump (op, left, right, name l1, name l1)`'.

`jump` As for `cjump`: '`jump (name l)`'.

`move` A `move` has always the form '`move (temp t, ...)`' or '`move (mem e, ...)`'.

2.2 Special Temporaries

Some of the temporaries have a special meaning for HAVM:

`fp`
`$fp` The frame pointer.

`sp`
`$sp` The stack pointer. One cannot read beyond `sp`.

`rv`
`$v0` The result register. Functions should store their result there. This is the strongest dependency on registers, since an expression such as '`1 + call (fact, 2)`' needs to "know" that the result of '`call (fact, 2`') is to be "read" in `rv`.

2.3 The HAVM Runtime

HAVM provides a set of predefined functions, modeled after the Tiger runtime.

`chr` (*code : int*) [string]
Return the one character long string containing the character which code is *code*. If *code* does not belong to the range [0..255], raise a runtime error: '`chr: character out of range`'.

`concat` (*first: string*, *second: string*) [string]
Concatenate *first* and *second*.

`exit` (*status: int*) [void]
Exit the program with exit code *status*. Note that contrary to SPIM 6.5, HAVM's own exit status is *status*.

`flush` () [void]
Flush the output buffer.

`getchar` () [string]
Read a character on input. Return an empty string on an end of file.

`init_array` (***size***: *int*, ***init***: *pointer*) [pointer]
Return the address of a freshly allocated array of *size* 4 byte elements, initialized to *init*.

`malloc` (***size***: *int*) [pointer]
Return the address of a freshly allocated block of memory of size *size*.

`not` (***boolean***: *int*) [int]
Return 1 if *boolean* = 0, else return 1.

`_not` (***boolean***: *int*) [int]
Same as `not`, but provided under two names so that people using `_not` in their MIPS output don't have to change the HAVM name. This is because the SPIM scanner cannot tell the difference between `not` as an instruction and as a label.

`ord` (***string***: *string*) [int]
Return the ascii code of the first character in *string* and -1 if the given string is empty.

`print` (***string***: *string*) [void]
Print *string* on the standard output.

`print_err` (***string***: *string*) [void]
Note: this is an EPITA extension. Same as `print`, but the output is written to the standard error.

`print_int` (***int***: *int*) [void]
`printint` (***int***: *int*) [void]
Note: this is an EPITA extension. Output *int* in its decimal canonical form (equivalent to '`%d`' for `printf`).

`size` (***string***: *string*) [int]
Return the size in characters of the *string*.

`strcmp` (***a***: *string*, ***b***: *string*) [int]
Note: this is an EPITA extension. Compare the strings *a* and *b*: return -1 if *a* < *b*, 0 if equal, and 1 otherwise.

`streq` (***a***: *string*, ***b***: *string*) [int]
`stringEqual` (***a***: *string*, ***b***: *string*) [int]
Note: this is an EPITA extension. Return 1 if the strings *a* and *b* are equal, 0 otherwise. Often faster than `strcmp` to test string equality.

`substring` (***string***: *string*, ***first***: *int*, ***length***: *int*) [string]
Return a string composed of the characters of *string* starting at the *first* character (0 being the origin), and composed of *length* characters (i.e., up to and including the character *first* + *length* - 1).

Let *size* be the size of the *string*, the following assertions must hold:

- 0 <= *first*
- 0 <= *length*
- *first* + *length* <= *size*

otherwise a runtime failure is raised: '`substring: arguments out of bounds`'.

2.4 LIR

A valid LIR program is a valid HIR program that in addition verifies the following constraints:

no nested `seq`
: The HIR tree must be flattened in a single thread of execution. Therefore, there must be at most one `seq` per function.

no `eseq`
: Similarly, the instruction `eseq` must not be used.

no nested `call`
: Calls cannot be embedded within other calls. Actually, the restriction is even stronger than this: a `call` can only appear in the following patterns:

  '`move` *`dest`* `call ...`'
  : A function call.

  '`sxp call ...`'
  : A procedure call.

one way `cjump`
: `cjump`s must be normalized in such a way that they are always followed by their negative destination.

3 Known Problems

Unfortunately, because HAVM is currently implemented with a naive recursive function, it is not robust to violations of the recursion due to `jump`s in high-level representation. Because in low-level intermediate representation there are no nested `jump`s, the problem does not arise.

This unique problem comes in several flavors, depending on how you generated the `jump`.

3.1 Ineffective break

The following example in Tiger uses a `break` *within an expression*.

```
while 1 do
  print_int ((break; 1))
```

The generated high level intermediate representation is:

```
/tmp % cat foo.hir
/* == High Level Intermediate representation. == */
# Routine: main
label main
# Prologue
# Body
seq
  label l1
  cjump ne
    const 1
    const 0
    name l2
    name l0
  label l2
  sxp
    call
      name print_int
      eseq
        jump  ###########   the break
          name l0
        const 1
    call end
  jump
    name l1
  label l0
seq end
# Epilogue
label end
```

If you run HAVM, it will loop.

The problem is the recursive evaluation: it recurses to evaluate the arguments of `print_int`, falls on the `jump`, executes it, and then finds the end of the control flow (`seq end`).

Then... it goes back to its previous recursive evaluation: that of the arguments of `print_int`, evaluates it, and honors the `jump` of the loop.

The traces demonstrate this:

```
 call ( name main ) []
7.2-7.10:  label l1
9.4-9.11:  const 1
10.4-10.11:  const 0
8.2-12.11:  cjump ne 1 0 ( name l2 ) ( name l0 )
17.6-20.15:  eseq
18.8-19.17:  jump ( name l0 )
20.8-20.15:  const 1
15.4-21.12:  call ( name print_int ) [1]
15.4-21.12:  end call ( name print_int ) [1] = ()
14.2-21.12:  sxp ()
22.2-23.11:  jump ( name l1 )
9.4-9.11:  const 1
10.4-10.11:  const 0
8.2-12.11:  cjump ne 1 0 ( name l2 ) ( name l0 )
17.6-20.15:  eseq
18.8-19.17:  jump ( name l0 )
20.8-20.15:  const 1
15.4-21.12:  call ( name print_int ) [1]
15.4-21.12:  end call ( name print_int ) [1] = ()
14.2-21.12:  sxp ()
22.2-23.11:  jump ( name l1 )
9.4-9.11:  const 1
10.4-10.11:  const 0
```

etc. etc.

3.2 Ineffective Boolean Operator

Another means to generate a `jump` that breaks the recursive evaluation is using optimized `if`. Consider the following Tiger code:

```
print_int (if 0 | 0 then 0 else 1)
```

which translates in the following High-Level Intermediate code:

```
3.0-3.10: label main
6.0-43.10: sxp
    7.2-43.10: call
        8.4-8.18: name print_int
        9.4-42.13: eseq
            10.4-41.11: seq
### The dispatching seq
                11.6-29.13: seq
                    12.8-16.17: cjump ne
                        13.10-13.17: const 0
                        14.10-14.17: const 0
```

```
                          15.10-15.17: name l3
                          16.10-16.17: name l4

                      17.8-17.16: label l3
                      18.8-22.17: cjump ne
                          19.10-19.17: const 1
                          20.10-20.17: const 0
                          21.10-21.17: name l0
                          22.10-22.17: name l1

                      23.8-23.16: label l4
                      24.8-28.17: cjump ne
                          25.10-25.17: const 0
                          26.10-26.17: const 0
                          27.10-27.17: name l0
                          28.10-28.17: name l1
                  11.6-29.13: seq end
### End of the dispatching seq
                  30.6-30.14: label l0
                  31.6-33.15: move
                      32.8-32.15: temp t0
                      33.8-33.15: const 0

                  34.6-35.15: jump
                      35.8-35.15: name l2
                  36.6-36.14: label l1
                  37.6-39.15: move
                      38.8-38.15: temp t0
                      39.8-39.15: const 1

                  40.6-40.14: label l2
              10.4-41.11: seq end
              42.6-42.13: temp t0
    7.2-43.10: call end
45.0-45.9: label end
```

Its verbose evaluation is:

```
% havm --trace /tmp/broken-if.hir; echo
plaining
unparsing
checking
checkingLow
evaling
 call ( name main ) []
9.4-42.13:  eseq
10.4-41.11:  seq
11.6-29.13:  seq
```

```
13.10-13.17:  const 0
14.10-14.17:  const 0
12.8-16.17:  cjump ne 0 0 ( name l3 ) ( name l4 )
25.10-25.17:  const 0
26.10-26.17:  const 0
24.8-28.17:  cjump ne 0 0 ( name l0 ) ( name l1 )
39.8-39.15:  const 1
37.6-39.15:  move ( temp t0 ) 1
40.6-40.14:  label l2
# End of the assignment seq.
11.6-29.13:  seq end
# Return to the dispatching seq.
30.6-30.14:  label l0
33.8-33.15:  const 0
31.6-33.15:  move ( temp t0 ) 0
34.6-35.15:  jump ( name l2 )
10.4-41.11:  seq end
42.6-42.13:  temp t0 = 0
7.2-43.10:  call ( name print_int ) [0]
7.2-43.10:  end call ( name print_int ) [0] = ()
6.0-43.10:  sxp ()
 end call ( name main ) [] = 0
0
```

If you read the example, you will notice that there are several `seq`s: the innermost branches to the label `l1`, then when it finishes the evaluation of the outer `seq`, it... returns to what its call-stack dictates: the evaluation of the inner `seq`. The latter drives it to execute the wrong assignment, producing an incorrect result.

Appendix A Copying This Manual

A.1 GNU Free Documentation License

0. PREAMBLE

   The purpose of this License is to make a manual, textbook, or other functional and useful document *free* in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

   This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

   We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

   This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

   A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

   A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The “Invariant Sections” are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The “Cover Texts” are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A “Transparent” copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not “Transparent” is called “Opaque”.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The “Title Page” means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, “Title Page” means the text near the most prominent appearance of the work’s title, preceding the beginning of the body of the text.

A section “Entitled XYZ” means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as “Acknowledgements”, “Dedications”, “Endorsements”, or “History”.) To “Preserve the Title” of such a section when you modify the Document means that it remains a section “Entitled XYZ” according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
D. Preserve all the copyright notices of the Document.
E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
H. Include an unaltered copy of this License.
I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

   You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

   The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

   In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements."

6. COLLECTIONS OF DOCUMENTS

   You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

   You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

   A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

   If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

   Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

   If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

   You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

   The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See `http://www.gnu.org/copyleft/`.

   Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

A.1.1 ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
Copyright (C)  year  your name.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.2
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.
A copy of the license is included in the section entitled ``GNU
Free Documentation License''.
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with...Texts." line with this:

```
with the Invariant Sections being list their titles, with
the Front-Cover Texts being list, and with the Back-Cover Texts
being list.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Index

F

H

L