1

Using the Assembler

This chapter describes how to run the **as** assembler, which produces an object file from one or more files of assembly language source code.

Note: Although **a.out** is the default file name that **as** gives to the object file that's created (as is conventional with most UNIX-style compiler systems), the format of the object file is not standard UNIX 4.3BSD **a.out** format. Object files produced by the assembler are in Mach-O (Mach object) file format. For more information about the Mach-O file format, see the *NEXTSTEP Development Tools and Techniques* manual.

Command Syntax

To run the assembler, type the following command in a shell window:

```
as [ option ] ... [ file ] ...
```

You can specify one or more command-line options. These assembler options are described in the following

section.

You can specify one or more files containing assembly language source code. If no files are specified, **as** uses the standard input (stdin) for the assembly source input.

Note: By convention, files containing assembly language source code should have a **.s** extension.

Assembler Options

The following command-line options are recognized by the assembler:

-o *name* The *name* argument after **-o** is used as the name of the **as** output file, instead of **a.out**.

-- Use the standard input (stdin) for the assembly source input.

-f Fast; no need to run **app** (the assembler preprocessor). This option is intended for use by compilers that produce assembly code in a strict ªcleanº format that specifies exactly where whitespace can go. The **app** preprocessor needs to be run on handwritten assembly files and on file that have been preprocessed by **cpp** (the C preprocessor). This typically is needed when assembler files are assembled through the use of the **cc**(1) command, which automatically runs the C preprocessor on assembly source files. The assembler preprocessor strips out excess spaces, turns each single-quoted character into a decimal constant, and turns occurrences of

```
# number filename level
```

into:

```
.line number;.file filename
```

The assembler preprocessor can also be turned off by starting the assembly file with **#NO_APP\n**.

When the assembler preprocessor has been turned off in this way, it can be turned on and off with pairs of **#APP\n** and **#NO_APP\n** at the beginning of lines. This is used by the compiler to wrap assembly statements produced from **asm**() statements.

-g Produce debugging information for the symbolic debugger **gdb**(1) so the the assembly source can be debugged symbolically. For include files (included by the C preprocessor's **#include** or by the assembler directive **.include**) that produce instructions in the (__TEXT,__text) section, the include file must be included while a **.text** directive is in effect (that is, there must be a **.text** directive before the include) and end with the a **.text** directive in effect (at the end of the include file). Otherwise the debugger will have trouble dealing with that assembly file.

-v Print the version of the assembler (both the NeXT version and the GNU version that it is based on).

-n Don't assume that the assembly file starts with a **.text** directive.

-I*dir* Add *dir* to the list of directories to search for files included with the **.include** directive. The default places to search are the current directory, and then **/usr/include**.

-L Save defined labels beginning with an `L' (the compiler generates these temporary labels). Temporary labels are normally discarded to save space in the resulting symbol table.

-W Suppress warnings.

Architecture Options

The program **/bin/as** is a driver that executes assemblers for specific target architectures. If no target architecture is specified, it defaults to the architecture of the host it is running on.

-arch *arch_type*
Specifies to the target architecture, *arch_type*, the assembler to be executed. The target assemblers

for each architecture are in **/lib/***arch_type***/as**.

-arch_multiple
This is used by the **cc**(1) driver program when it is run with multiple **-arch** *arch_type* flags and instructs programs like **as**(1) that if it prints any messages to precede the messages with one line stating the program nameÐin this case **as**Ðand the architecture (from the **-arch** *arch_type* flag) to distinguish which architecture the error messages refer to. This flag is accepted only by the actual assemblers (in **/lib/***arch_type***/as**) and not by the assembler driver, **/bin/as**.

M68000-Specific Options

-l For offsets from an address register that refers to an undefined symbol (as in **a6@(var**) where **var** is not defined in the assembly file), make the offset and the relocation entry width 32 bits rather than 16 bits.

-k Produce a warning when a statement of the form

```
.word symbol1-symbol2+offset
```

does not fit in a 16-bit word. This is only applicable on the 68000 processor, where **.word** is 16 bits and all addresses are 16 bits; therefore, this option isn't applicable on NeXT computers.

-mc68000 and **-mc68010**
Don't generate branches that use 32-bit **pc**-relative displacements (which aren't implemented on the 68000 and 68010 processors). These options aren't applicable on NeXT computers.

-mc68020 Generate branches that use 32-bit **pc**-relative displacements. This is the default behavior.