7

PA-RISC Assembler Instructions

This chapter contains information specific to the assembler instruction set for the PA-RISC processor architecture. Because NEXTSTEP's assembler instruction set for PA-RISC is almost identical to that for the assembler on Hewlett Packard's HPUX, this chapter lists the differences and refers you to these Hewlett Packard publications:

- *PA-RISC 1.1 Architecture and Instruction Set Reference Manual*, Third Edition, HP Part Number 09740-90039.
- *PA-RISC Procedure Calling Conventions Reference Manual*, Second Edition, HP Part Number 09740-96015.

PA-RISC   Assembler-Instruction Differences

The following sections describe the differences the developer in syntax and directives between NEXTSTEP's assembly code for PA-RISC and HPUX assembler. This information is of use to those who write assembly code for PA-RISC and to those who are porting existing assembly modules to NEXSTEP.

Syntatical Differences

- **Registers**:   Use the percentage sign (%) to denote a register.   Some examples of register designation are:

%r1, **%sr1**, **%fp1**, and **%cr1**.

- **Effective Address**: The effective address is usually specified as *d*(*s*,*b*) where the base register *b* and the displacement *d* form the offset. The space identifier is *s*. If *s* is zero, you can specify *d*(*b*) in place of *d*(0, *b*).
- **Labels**: The NEXTSTEP assembler requires a colon after the label where it is declared while HPUX does not. For example.

```
HPUX                   NEXTSTEP
   bl  there, %r2            bl  there, %r2
there                  there:
   nop                 nop
```

- **Comments**: To specify comments starting at the beginning of a line, use the hash mark character (#). To specify comments in-line, use a semicolon (;). For example:

```
# a comment from the beginning of the line
; this is a valid comment, too
   €€€;and so is this
   nop  ;and this is also a comment
```

- **Instruction Separator**: Use the at-sign (@) to separate multiple instructions on the same line (a feature normally found in macros). For example:

```
nop @ nop
```

- **Field Selectors**: The NEXTSTEP assembler only supports the field selectors L` and R` (note that the quote character is a back quote). Examples of their usage are:

```
ldil        L`_printf, %r2
ble    R`_printf(%sr0, %r2)
```

Differences in Directives

The table below lists the directives that have equivalent (or nearly equivalent) forms. Hewlett Packard directives that are not on the list do not have NEXTSTEP equivalents; however, you can implement these as macros or pseudo-ops.

HPUX	**NEXTSTEP**
.ALIGN	.align
.COMM	.comm
.ENDM	.endmacro
.EQU	.set or *name* = *expression*
.EXPORT	.globl
.MACRO	.macro
.ORIGIN	.org
.BLOCK	.space
.BLOCKZ	.space
.BYTE	.byte
.DOUBLE	.double
.FLOAT	.single
.HALF	.short
.STRING	.ascii
.STRINGZ	.asciiz
.WORD	.long

In addition, the directives for changing spaces and subspaces to standard ones have equivalents; these map conceptually to Mach-O segments and sections.

HPUX	**NEXTSTEP**
.CODE	.text
.DATA	.data

The HPUX assembler allows you to declare arbitrary spaces and subspaces. The NEXTSTEP assembler has a predefined list of segments and sections with their own special directives. (See <x-ref> earlier in this manual for more information.)

Pseudo-Ops

Simple HPUX pseudo-ops, such as COPY, exist as predefined macros in NEXTSTEP assembler code. More complicated HPUX pseudo-ops, such as ENTER and LEAVE, are implemented as macros. To make use of these macros, compile the assembly code with **cc** and not **as** so that you can take advantage of the C preprocessor. The file **/NextDeveloper/Headers/architecture/hppa/asm-help.h** contains definitions of these macros.