Pseudo Instructions [5]

Pseudo instructions direct the assembler in its task of interpreting source statements and generating an object program.

> **Note:** A detailed description of the pseudo instructions presented in this section are listed in alphabetical order in appendix A, page 189.

Each program module begins with an IDENT pseudo instruction and ends with an END pseudo instruction. Symbol, micro, macro, and opdef definitions that occur within the program module are cleared before assembling the next program module.

Definitions of symbols, micros, macros, or opdefs included before the first IDENT pseudo instruction or between an END and a subsequent IDENT pseudo instruction are global and can be referenced in any subsequent program module (see subsection 3.1.2, page 35).

Redefinable micros and symbols can only be defined locally. If they appear before the first IDENT or between an END and subsequent IDENT pseudo instruction they are cleared after assembling the next program module.

Symbolic machine instructions and the following pseudo instructions must appear within a program module. They are allowed outside of an IDENT to END sequence only within opdef or macro definitions.

ALIGN	BSS	CON	LOC	START
BITP	BSSZ	DATA	ORG	VWD
BITW	COMMENT	ENTRY	QUAL	
BLOCK	COMMON	EXT	SECTION	

The LOCAL pseudo instruction must occur immediately after a macro or opdef prototype statement or after a DUP or ECHO pseudo instruction. Comment statements can intervene. All other pseudo instructions, macro definitions, and opdef definitions can appear anywhere in a CAL program.

Pseudo instructions are classified and described according to their applications, as follows:

Class	Pseudo instructions
Program control	IDENT, END, COMMENT
Loader linkage	ENTRY, EXT, START
Mode control	BASE, QUAL, EDIT, FORMAT
Section control	SECTION, BLOCK, COMMON, STACK, ORG, LOC, BITW, BITP, BSS, ALIGN
Message control	ERROR, ERRIF, MLEVEL, DMSG
Listing control	LIST, SPACE, EJECT, TITLE, SUBTITLE, TEXT, ENDTEXT
Symbol definition	=, SET, MICSIZE, DBSM
Data definition	CON, BSSZ, DATA, VWD
Conditional assembly	IFA, IFC, IFE, IFM, SKIP, ENDIF, ELSE
Micro definition	CMICRO, MICRO, OCTMIC, DECMIC
File control	INCLUDE
Defined sequences	MACRO, OPDEF, DUP, ECHO, ENDM, ENDDUP, STOPDUP, LOCAL, OPSYN, EXITM, NEXTDUP

Note: You can specify pseudo instructions in uppercase or lowercase, but not in mixed case.

Program control
5.1

The program control pseudo instructions define the limits of a program module and include the following:

Pseudo	Description
IDENT	Marks the beginning of a program module.
END	Marks the end of a program module.
COMMENT	Enters comment, generally a copyright, into the generated binary load module.

Loader linkage
5.2

The loader linkage pseudo instructions provide for the loading of multiple object program modules, linking them into one executable program (ENTRY and EXT), and specifying the main program entry (START).

The loader linkage pseudo instructions include the following:

Pseudo	Description
ENTRY	Specifies symbols, defined as addresses or values, so that they can be used by other program modules linked by a loader.
EXT	Specifies linkage to addresses or values defined as entry symbols in other program modules.
START	Specifies symbolic address at which execution begins.

Mode control
5.3

Mode control pseudo instructions define the characteristics of an assembly. The BASE pseudo instruction determines whether notation for numeric data is assumed to be octal or decimal. The QUAL pseudo instruction permits symbols to be defined as qualified or unqualified. The EDIT pseudo instruction controls the editing of assembler statements. The FORMAT pseudo instruction controls the format that is used for interpreting assembly source statements.

The mode control pseudo instructions include the following:

Pseudo	Description
BASE	Specifies data as being octal, decimal, or a mixture of both.
QUAL	Designates a sequence of code where symbols may be defined with a qualifier, such as a common routine with its own labels.
EDIT	Turns editing on or off.
FORMAT	Changes the format to old or new.

Section control 5.4

Section control pseudo instructions control the use of sections and counters in a CAL program.

The section control pseudo instructions include the following:

Pseudo	Description
SECTION	Defines specific program sections and replaces the BLOCK and COMMON pseudo instructions. The SECTION pseudo instruction is recommended for use with all Cray PVP systems because it includes all of the capabilities of BLOCK and COMMON pseudo instructions.
BLOCK	Defines local sections.
COMMON	Defines common sections that can be referenced by another program module.
STACK	Increments the size of the stack.
ORG	Resets location and origin counters.
LOC	Resets location counter.
BITW	Sets the current bit position relative to the current word.
BITP	Sets the current bit position relative to the current parcel.

Pseudo	Description
BSS	Reserves memory.
ALIGN	Aligns code on an instruction buffer boundary.

Message control
5.5

Two pseudo instructions, ERROR and ERRIF, let you generate an assembly error condition. The MLEVEL pseudo instruction lets you change the level of messages you receive in your source program.

Pseudo	Description
ERROR	Sets an assembly error flag
ERRIF	Sets an assembly error flag according to the conditions being tested
MLEVEL	Sets the level at which messages are reported in the source listing
DMSG	Issues a comment-level message containing the string found in the operand field

Listing control
5.6

Listing control pseudo instructions control the content and format of the listing produced by the assembler. These pseudo instructions are not listed unless the LIST pseudo instruction is specified by using the LIS option.

The listing control pseudo instructions are as follows:

Pseudo	Description
LIST	Controls listing by specifying particular listing features that will be enabled or disabled
SPACE	Inserts blank lines in listing
EJECT	Begins new page
TITLE	Prints main title on each page of listing
SUBTITLE	Prints subtitle on each page of listing

Pseudo	Description
TEXT	Declares beginning of global text source
ENDTEXT	Terminates global text source

Symbol definition
5.7

The =, SET, and MICSIZE pseudo instructions define symbols used in the program. Requirements for symbols are specified in subsection 4.3, page 69. The symbol definition pseudo instructions are as follows:

Pseudo	Description
=	Equates a symbol to a value; not redefinable.
SET	Sets a symbol to a value; redefinable.
MICSIZE	Equates a symbol to a value equal to the number of characters in micro string; redefinable.
DBSM	Generates a named label entry in the debug symbol tables with a specific type specified.

Data definition
5.8

Data definition pseudo instructions are the only pseudo instructions that generate object binary. The only other instructions that are translated into object binary are the symbolic machine instructions. An instruction that generates binary cannot be used with a section that does not allow instructions, data, or both.

The data definition pseudo instructions are as follows:

Pseudo	Description
CON	Places an expression value into one or more words
BSSZ	Generates words that have been initialized to 0
DATA	Generates one or more words of numeric or character data

Pseudo	Description
VWD	Generates a variable-width field of word-oriented data

Conditional assembly
5.9

The conditional assembly pseudo instructions permit the optional assembly or skipping of source code. The conditional pseudo instructions IFA, IFC, or IFE determine whether the sequence of instructions following the test will be skipped or assembled. The end of the conditional sequence is determined by a count of instructions provided in the test instruction or by an ENDIF pseudo instruction with a matching location field name.

The ELSE pseudo instruction provides a means of reversing the effect of a previous IFA, IFE, IFC, SKIP, or ELSE instruction. The SKIP pseudo instruction unconditionally skips the statements that follow it.

When skipping under the control of a statement count, comment statements (denoted by an asterisk (*) in column 1) and continued lines are not included in the statement count.

When an IFA, IFE, IFC, SKIP, or ELSE pseudo instruction initiates skipping, editing is disabled. When the skip sequence is completed, the assembler returns to the editing mode in effect before skipping was initiated.

To specify a conditional assembly, use the following pseudo instructions:

Pseudo	Description
IFA	Tests expression attributes; address and relative attributes.
IFE	Tests two expressions for some assembly condition; less than, greater than, and equal to.
IFC	Tests two character strings for assembly condition; less than, greater than, and equal to.
IFM	Test for machine characteristics.
SKIP	Unconditionally skip subsequent statements.

Pseudo	Description
ENDIF	Terminates conditional code sequence.
ELSE	Reverses assembly condition.

Micro definition

5.10

Through the use of micros, programmers can assign a name to a character string and subsequently refer to the character string by its name. A reference to a micro results in the character string being substituted for the name before assembly of the source statement containing the reference.

The following pseudo instructions specify micro definition:

Pseudo	Description
CMICRO	Constant micro; assigns a name to a character string.
MICRO	Redefinable micro; assigns a name to a character string.
OCTMIC	Converts the octal value of an expression to a character string and assigns it a redefinable name.
DECMIC	Converts the decimal value of an expression to a character string and assigns it a redefinable micro name.

In addition to the micros previously listed, the CAL assembler provides predefined micros. They can be specified in all uppercase or all lowercase, but not mixed case. CAL provides the following predefined micros:

Micro	Description
$DATE	Current date – *'mm/dd/yy'*
$JDATE	Julian date – *'yyddd'*
$TIME	Time of day – *'hh:mm:ss'*
$MIC	Micro character – double quotation mark (")

Micro	Description
$CNC	Concatenation character – underscore (_).
$QUAL	Name of qualifier in effect; if none, null string.
$CPU	Target machine: 'CRAY YMP', 'CRAY C90', 'CRAY J90', or 'CRAY TS'.
$CMNT	Comment character used with the new format – semicolon (;).
$APP	Append character used with the new format – circumflex (^).
AREGSIZE	Number of bits in an A register of the current target machine. For CRAY C90, CRAY J90, or CRAY Y-MP systems, AREGSIZE = 32. For CRAY T90 systems, AREGSIZE = 64.
PREGSIZE	Number of bits in the Program register of the current target machine. For CRAY J90 and CRAY Y-MP systems, PREGSIZE = 24. For CRAY C90 systems, PREGSIZE = 32. For CRAY T90 systems, PREGSIZE = 32

The following example illustrates the use of a predefined micro ($DATE):

```
      DATA   'THE DATE IS "$DATE"'
      DATA   'THE DATE IS 06/23/94'†
```

You can reference micro definitions anywhere in a source statement, except in a comment, by enclosing the micro name in quotation marks. If column 72 of a line is exceeded because of a micro substitution, the assembler creates additional continuation lines. No replacement occurs if the micro name is unknown or if one of the micro marks was omitted.

In the following example, a micro called PFX is defined as the character string ID. A reference to PFX is in the location field of a line.

```
"PFX"TAG   S0            S1    ; Left-shifted three spaces when edited.
```

In the following example, before the line is interpreted, CAL substitutes the definition for PFX producing the following line:

```
IDTAG    S0              S1    ; Left-shifted three spaces when edited.
```

The following example shows the use of the predefined micros, AREGSIZE and PREGSIZE:

```
A    =    "AREGSIZE"          ; Size of the A registers.
     CON  A                   ; Store value in memory.
B    =    "PREGSIZE"          ; Size of the Program register.
     CON  B                   ; Store value in memory.
```

File control

5.11

The file control psuedo instruction, INCLUDE, inserts a file at the current source position. The INCLUDE pseudo instruction always prepares the file for reading by opening it and positioning the pointer at the beginning.

You can use this pseudo instruction to include the same file more than once within a particular file.

You can also nest INCLUDE instructions. Because you cannot use INCLUDE recursively, you should review nested INCLUDE instructions for recursive calls to a file that you have already opened.