Printer Ruby

Spruce version 12.0 -- spooler version 12.0

File: asm.tty

Creation date: 28-Sep-82 10:21:02

Printing date: 28-Sep-82 10:21:57 EDT

For: slomer

8 total sheets = 7 pages, 1 copy.

ASM

This assembler, written in BCPL, runs on the Alto and produces BCPL-compatible relocatable binary output files, suitable for input to BLDR, the BCPL loader. The Alto Hardware manual describes the source language and the virtual machine.

1. Symbols

Symbols may be up to 130 characters in length, and every character of a symbol must be used to identify it. By default upper- and lower-case characters are different, and two character strings represent the same symbol only if the same letters and cases are used in both. However, the /U switch causes all lower-case letters in symbols to be changed to upper case (even in external symbols). Thus if you want an assembly-language program to link to symbols containing lower-case letters, you must either default lower-case conversion in ASM or map all symbols to upper case in BLDR using its /U switch.

2. Strings

Strings follow BCPL conventions. They may not extend from one line to the next.

3. Assembly Regions

This assembler can assemble into three regions: two static regions (one in page 0) and one code region. The directives .NREL, .SREL, and .ZREL cause the assembler to begin placing code in the code region, the non-page-0 static region, and the page 0 static region, respectively. The BCPL loader causes the restrictions that the code area may not contain pointers into the code area, that the first word of the code area may not point to a static area, and that no static area may contain pointers to a static area. The only external symbols are statics.

Arithmetic is not allowed on symbols denoting statics, and the symbol "." is undefined in .SREL and .ZREL. Any absolute or code- relative expression (including such goodies as JMP@ 62) may be placed in .SREL or .ZREL. Any absolute expression, static reference, or instruction reference to .ZREL may appear in .NREL.

---

4. Text

There are two text modes, .TXTM B and .TXTM L. Mode B causes the generation of standard BCPL strings. Mode L causes the generation of long strings, a full word length followed by the string characters, two per word.

5. .GET

The directive .GET "FOO" causes the file FOO to be inserted into the source text at that point. .GET can be used up to two levels deep. Its primary utility is likely to be for lists of externals and for canned entry and exit sequences.

6. .GETNOLIST

Works exactly like .GET, except that the "gotten" file is not included in the listing, nor are any files which it .GET's.

7. .BEXT

In addition to .EXTN and .EXTD and .ENT, I have added two directives .BEXT and .BEXTZ which work exactly as BCPL's External works for non-page-0 and page 0 statics, respectively. This should increase the utility of the .GET feature above.

8. Expressions

Parentheses (but not precedence) are supported. Constructs like "K and $*N and 5 and 17. and 3B10 are all primaries. Most BCPL and customary assembler operators are allowed. The construct 1B10 means 40(octal), unlike BCPL's convention. I am willing to be convinced on this point.

9. Predefined Symbols

All predefined symbols and directives and opcodes are defined both in all upper-case and all lower-case letters. For example, both LDA and lda are predefined, but Lda is not. The following Alto-specific opcodes are preloaded in the symbol table:

JSRII	JSRIS	CYCLE	CONVERT	DIR	EIR	BRI
RCLK	SIO	BLT	BLKS	SIT	RDRM	WTRM
JMPRM	MUL	DIV				

In addition, the following pile of skips which test various conditions has been added, courtesy of Dan Ingalls. Only the names have been changed to confuse the innocent:

Two operands:

SZE	SZ	SNZ	SP	SGZ	SN	SEQ
SE	SNE	SLT	SLE	SGT	SGE	SGTU
SLEU	SGEU	SLTU	SODD	SKEVEN	SNIL	SNNIL
MKZERO	MKONE	MKNIL	MKMINUSONE			

No Operands:

NOP	SKIP

It should be explained that U stands for unsigned, and that Dan thinks of NIL as -1.

10. Operation

If the source file is called FOO.ASM, type

```
ASM FOO.ASM
```

If you just type ASM FOO it will first try to use FOO and, failing in that, try FOO.ASM. The assembler will usually want to construct several files, which it will do by substituting various extensions on FOO unless you specify otherwise. There are a lot of switches which apply to ASM:

/L Construct a listing file
/S Include the symbols defined by the user, for what they're worth
/A Include all symbols, even the predefined ones
/R Include a printout of the .BR file
/N Don't make a .BR file
/E Make an .ER file which is a copy of the error messages sent to the terminal
/D Print debugging messages (as errors, in fact)
/P Pause after printing each error message (continue with CR)
/U Map all lower-case letters in symbols to upper-case

There are also a lot of switches which apply to file names, and which tell the assembler to use this name instead of the one it was about to invent:

/L Names the listing file
/E Names the error file
/S Names the source file (also no switches)
/T Names the temporary file
/B Names the relocatable binary file