

# 32000/ASSEMBLER



REFERENCE  
MANUAL



# 32000 ASSEMBLER



PART NO 0410,005  
ISSUE NO 1  
JULY 1985

© Copyright Acorn Computers Limited 1985

Neither the whole or any part of the information contained in, or the product described in, this manual may be reproduced in any material form except with the prior written approval of Acorn Computers Limited (Acorn Computers).

The product described in this manual and products for use with it, are subject to continuous developments and improvement. All information of a technical nature and particulars of the product and its use (including the information in this manual) are given by Acorn Computers in good faith.

In case of difficulty please contact your supplier. Deficiencies in software and documentation should be notified in writing, using the Acorn Scientific Fault Report Form to the following address:

Sales Department  
Scientific Division  
Acorn Computers Ltd  
Fulbourn Road  
Cherry Hinton  
Cambridge  
CB1 4JN

All maintenance and service on the product must be carried out by Acorn Computers' authorised agents. Acorn Computers can accept no liability whatsoever for any loss or damage caused by service or maintenance by unauthorised personnel. This manual is intended only to assist the reader in the use of the product, and therefore Acorn Computers shall not be liable for any loss or damage whatsoever arising from the use of any information or particulars in, or any error or omission in, this manual, or any incorrect use of the product.

Published by Acorn Computers Limited,  
Fulbourn Road, Cherry Hinton, Cambridge CB1 4JN.

Within this publication the term BBC is used as an abbreviation for the British Broadcasting Corporation.

**NOTE:** A User Registration Card is supplied with the hardware. It is in your interest to complete and return the card. Please notify Acorn Scientific at the above address if this card is missing.

ISBN 0 907876 36 6 Acorn Scientific

# Contents

1	Introducing the Acorn 32000 assembler	1
1.1	Installation	2
1.2	Assembler commands	2
1.3	Assembler options	3
1.4	Assembler listing format	4
2	32000 assembler source format	7
2.1	Format of source lines	7
2.2	Character set	8
2.3	Symbols	8
2.4	Constants	9
2.5	Expressions	10
2.6	Mnemonic conventions	11
3	Assembler directives	13
3.1	Absolute and relocatable modes	13
3.2	Standard directives	13
3.2.1	GET	14
3.2.2	CHAIN	14
3.2.3	EQU	14
3.2.4	EQUR	15
3.2.5	SET	15
3.2.6	NLSYM	15
3.2.7	IF	16
3.2.8	IFDEF and IFNDEF	16
3.2.9	MACRO...MEND	17
3.2.10	DCB	21
3.2.11	DCS	21
3.2.12	DCW	21
3.2.13	DCD	21
3.2.14	DCF	22
3.2.15	DCL	22
3.2.16	ALLOCB	22
3.2.17	ALLOCW	22
3.2.18	ALLOCD	22
3.2.19	ALIGN	23
3.2.20	ENTRY	23
3.2.21	END	23

3.2.22	TITLE	24
3.2.23	OPTIONS	24
3.3	Object module directives	25
3.3.1	MODULE	25
3.3.2	AREADEF	25
3.3.3	AREA	27
3.3.4	AREALEN	27
3.3.5	AREAEND	28
3.3.6	EXPORT and EXPORTC	28
3.3.7	IMPORTC	28
3.3.8	IMPORT	29
3.3.9	HANDLER	29
3.3.10	SPECSB and DEFSB	29
3.3.11	ADDRESS	30
3.3.12	CDESC	30
3.3.13	LINKNO	31
3.4	Treatment of labels	31

# 1 Introducing the Acorn 32000 assembler

This document is a reference guide to the Acorn 32000 macro assembler. It is not a tutorial guide, and therefore the reader is assumed to be familiar with:

32000 Assembly language.

This is described in the *Instruction Set Reference Manual* which is available from dealers. Although the mnemonics used by the assembler are to the National standard, pseudo-operations (assembler directives) are specific to the Acorn Assembler.

The Panos operating system.

The use of the operating system environment under which the assembler runs is described in a document supplied with the system, the *Panos Guide to Operations* and the *Panos Programmer's Reference Manual*. The user is also assumed to know how to use the command to call the assembler.

Acorn Object Format (AOF)

is mentioned frequently in this guide; the output produced by the assembler will usually be in this form, although knowledge of the details is unlikely to be required. A full description of AOF is given in the *Panos Technical Reference Manual*. See also the various User Guides for introductory material.

Features of the assembler include:

- complete support of the NS32000 instruction set including Memory Management and Floating Point extensions.
- support of all nine categories of the general addressing modes of the NS32000.
- two types of object file:
  1. an image in Acorn Object Format suitable for linking into a Panos relocatable image using the system linker.
  2. a simple binary image suitable for immediate execution from the Pandora \* prompt.
- powerful macro defining capability. The user may define macro instructions in the source which may be called to insert common

sequences of 32000 mnemonics or assembler directives. Macros may call other macros, and recursion is possible.

- conditional assembly. The ability to assemble parts of the source conditionally is made even more useful by the ability to set 'flag' symbols on the command line so that different versions may be assembled from the same source file.

## 1.1 Installation

The assembler is supplied on a 5¼ inch floppy disc in Acorn DFS format. This needs to be installed even if it is intended for use in conjunction with the DFS. Refer to the appropriate User Guide supplied with the hardware for details about installing the assembler.

## 1.2 Assembler commands

This section summarises the arguments of the assembler command. See the beginning of chapter 2 for a breakdown of the metasyntax used here.

{-source} filename (-asm)

This names the source file to be assembled. The extension '-asm' will be appended if no other extension is given. Multiple files may be assembled using the CHAIN directive.

-list {name}

An assembly listing may be sent to a file called source-lis, or to another named file or device. The format of the listing is described in section 1.4. See figure 1 for a demonstration of this option.

-error {name}

Assembly errors are reported to the initial error stream by default (i.e. the messages usually go to the screen). The 'error' argument names an alternative destination for errors.

-aof {filename}

The output from the assembler is put into a file source-aof by default. The 'aof' argument allows an alternative file to be named. Note that the file may not in fact be an AOF file, if the assembly was carried out in absolute or relative binary mode.

**-opt options**

Several options are provided to change the behaviour of the assembler. These are described in section 1.3 below.

**-get "mapping {, mapping}\*"**

This argument is used to specify a mapping between filenames specified in GET and CHAIN directives, and the actual filename to be used. The word 'get' is followed by a string in double quotes which is a comma-separated list of mappings from GET (or CHAIN) names to filenames. For example:

```
-> asm32 -source fred -get "fpStuff=fp-asm,debug=db-asm"
```

With this mapping, a "GET fpStuff" directive would access file fp-asm.

**-identify**

Specifying this argument causes the assembler to print its version number.

**-help-**

Specifying this argument causes the assembler to produce a summary of the arguments which may appear on the command line.

### 1.3 Assembler options

The -opt argument is followed by a list of letters which are used to flag various options. A flag letter preceded by a '+' enables the option; a '-' sign disables. The exception is '\$', which is followed by the name of the symbol to be set or reset. Note that if the first option letter is preceded by a '-', then the whole option string must be enclosed in double quotes e.g. -opt "-l-m".

- c Usually upper and lower case are treated as distinct characters in identifiers. Quoting opt +c causes cases to be equated upon reading each source line, so that fred and FRED are the same symbol.
- l Usually source files are loaded into store during the first pass (if there is enough space) to minimise disc accesses on subsequent passes. This occasionally causes the assembler to run out of room. Quoting opt -l will disable loading and thus prevent the no room error (unless there genuinely isn't enough memory for the assembly).

- m By default the assembler will try to optimise the size of the output file by taking many passes over the source. Giving the `opt -m` option causes the assembler to make only enough passes to resolve symbol references, at the expense of producing non-optimally sized output code. This option only applies when absolute binary rather than AOF is generated.
- p Usually the assembler produces 'packed' style AOF files. Quoting `opt -p` causes general format AOF files to be produced.
- \$ This option is followed by a name to be set to TRUE. This name may be accessed in a conditional assembly ( IF) directive in the source. For example, `-opt $debug` sets the symbol 'debug' to TRUE (-1). Following the name by a single quote, e.g. `-opt $debug'` sets the symbol to FALSE (0).

As implied by the descriptions above, the default state of the options is:  
+LMP-C.

## 1.4 Assembler listing format

An assembly listing is produced if required by giving the `-list` argument on the command line. It has the following format:

```
lllll b1 b2 b3 b4 b5 b6 nnnn text.....
```

where:

lllll

is the value of the location counter at the start of the code for the line, printed as a 6-digit hex number.

nnnn

is the source line number.

b1..b6

are the byte values (in hex) of the generated codes. b1 is at the lowest address. Spaces are printed if less than 6 bytes were generated;

Extra bytes are displayed on following lines in the form:

```
lllll b7 b8 b9 etc.
```

with at most 6 bytes per line, and llllll being the address of the first byte on each extra line. Lines which came from a macro expansion in the source are marked with a + character at the start of the line.

At the end of the assembly, the assembler sends the following statistics to the output stream, which is the vdu by default (only if the global string Program\$Verbosity is set to greater than 1 - see the *Panos Guide to Operations*):

- The number of errors detected.
- The total size (in bytes) of the area(s).
- The number of passes required

Incorrect lines are echoed to both the listing file and the error file. Errors are reported using textual messages printed out before the failing line. Figure 1 gives an example of an error message from an assembly within the Panos editor. The source can be seen in the background, the assembly command appearing in the top window, with the error message contained in the lower window.

```

Press SHIFT with ESCAPE for Help                                16 Aug 85 15:09:31
-> Rom32 asrat -list asrat-lis

MOVW0 0, R0
LXP 0

Msg1a DCS "Type limit for prime numbers (2 to "
Msg1b DCS "): "
Msg1a DCS "Primes from 2 to "
Msg1b DCS "are:"

; subroutines

; ReadInt

0000C1 03 29 30 20      141 Msg1b DCS "): "
** Error: undefined symbol used (Primes)
** Error: superfluous characters following instruction or directive
0000C2 03 20 61 72 65 30 143 Msg1a DCS "Primes from 2 to "
0000C3 00 20 61 72 65 30 143 Msg1b DCS "are:"
  
```

Figure 1 Assembler error message



## 2 32000 assembler source format

The Acorn 32000 assembler accepts standard National Semiconductor instruction mnemonics, and in addition provides a full set of pseudo-mnemonics (assembler directives) and the ability to define macro instructions.

A source program is a sequence of lines which may contain 32000 assembly language mnemonics, assembler directives, comments, or nothing at all.

Within this document a meta-syntax is used to describe the syntax of assembler source lines. In this meta-syntax, the characters {, }, |, \* and ' have special meanings:

{x}	means 0 or 1 occurrences of x
{x}*	means 0 or more occurrences of x
{x y}	means 1 occurrence of x or 1 occurrence of y
'c'	where c is a single reserved character, means the literal character c, i.e. any special meaning is disabled. If c is not a single character or not a reserved character then ' stands for itself.
name	is a syntax class-name (i.e. lower case text. Upper case text is used for literal items, e.g. MOVQD, END).

All other symbols stand for themselves.

### 2.1 Format of source lines

The format of a source line is:

```
{label} {mnemonic {operand [,operand]*}} {;comment}
```

If a label is present, it must start at the beginning of a source line. Any mnemonic must be preceded by at least one space. A comment may start at any position on the line; it is marked by a semi-colon and continues up to the end of the line. There must be at least one space between a mnemonic and any following operands, but no space need precede a comment.

Operands are separated by commas and may contain spaces. These are ignored, except within string constants. Expressions are therefore allowed to

contain blanks, which are ignored. However, spaces are not allowed in tags (see later), numeric constants, and compound symbols such as `> =`.

A source line may contain up to 255 characters. The assembler will stop with an error if more than this number of characters occur without a line-break, since this would suggest an erroneous source file, e.g. a file which is not a text file.

## 2.2 Character set

The character set consists of letters (upper and lower case), digits, the underscore character (`_`) and other special characters. Upper and lower case are distinct, except in instruction mnemonics, directives, and macro names. The use of option `c` will cause the assembler to equate upper and lower case in identifiers.

## 2.3 Symbols

Symbols consist of letters, digits and underscores, starting with a letter or underscore. Symbols are significant to 63 characters.

A relocatable symbol is one which is defined as a label in a relocatable area. All other symbols are either absolute or external.

Some symbols are reserved and so cannot be redefined. These are:

```
R0, R1, .. R7, F0, F1, .. F7
TOS, EXTERNAL, FP, SP, SB, PC
The MMU registers
```

Mnemonics, e.g. `END` and `MOVQB`, are allowed as label names however, so lines such as:

```
END END
```

are allowed but not advisable.

Note that the letters used in the option field of the string instructions (`MOVSi`, `CMPSi` etc.) and the `SETCFG` instruction are not reserved; they are marked by the fact that they appear in this specific context (inside square brackets).

## 2.4 Constants

Integers may be given as unsigned decimal numbers or in the forms #Xhhhh, #Bbbbb, #Odddd, for hexadecimal, binary and octal representations respectively. Note that the letters A through F used in hex numbers may occur in either upper or lower case. An alternative representation of hexadecimal numbers is the form :hhhh .

All integers are interpreted as 32-bit quantities.

Floating point constants are optionally signed and have an optional exponent. Examples are:

```
100
1.1
-.1
-1e4
1.234E-1
```

Floating point constants are allowed only in the DCF and DCL directives, and as floating point immediate operands.

String constants are delimited by single quotes in the simple case, and double quotes to generate a counted-string form, i.e. the bytes in the text preceded by a byte containing the length of the text. The DCB directive (described later) accepts either form, creating the appropriate stream of bytes.

Character constants are also valid in integer expressions, where their length is limited to 4 characters in the simple form, and 3 characters in the counted-string form, the length byte forming part of the value in the latter case.

The value of a multi-character constant as an integer is calculated using the same store interpretation adopted by the 32000 architecture, i.e. the least significant byte is the byte at the lowest address, which is the leftmost character of a string, or the length byte in counted strings.

Hence:

```
'A' = #X41, "A" = #X4101, 'AB' = #X4241, "AB" = #X424102 etc.
```

Within string constants, the asterisk \* is used as an escape character. If the character that follows is an N or n, then the actual byte value stored at the

current position is determined by the value of the NLSYM option (see directive descriptions later); the default value is 10 (ASCII LF=NL).

If the following one or two characters are valid hex digits, then the number they represent is planted as a byte value. This enables the simple insertion of control characters within strings. For example `A*N` generates the bytes `#X41,#X0A`; `*03*FEA` generates `#X03,#XFE,#X41`. If neither of these cases holds, the following character is planted without interpretation; hence a single asterisk is represented in a string as `**`. Because this mechanism allows the representation of the newline character in strings, it is forbidden for strings to cross line boundaries.

The special symbol '\$' is used to stand for the program counter. For example:

```
BR $      ;infinite loop
```

## 2.5 Expressions

All expressions are calculated to 32 bits and overflow is ignored. Evaluation is ordered according to the priority below, and left-to-right for operators of the same precedence. Bracketed sub-expressions are evaluated first. The arithmetic and comparison operators treat their operands as signed quantities; the 6 operators in the latter group return TRUE (-1) or FALSE (0).

### Operator Priority Functions

-	8	Unary minus
~	8	Bitwise complement (unary)
<<	7	Logical left shift (0s shifted in from right)
>>	7	Logical right shift (0s shifted in from left)
&	6	Bitwise AND
	6	Bitwise OR
~	6	Bitwise exclusive OR
*	5	Multiply
/	5	Divide (as defined by QUOD instruction)
%	5	Remainder (Modulus - as defined by REMD instruction)
-	4	Subtract
+	4	Add

=	3	Equal-to
<>	3	Not-equal-to
<	3	Less-than *
>	3	Greater-than *
<=	3	Less-than-or-equal-to *
>=	3	Greater-than-or-equal-to *
!	2	Conditional NOT
&&	1	Conditional AND
	1	Conditional OR

The comparison operators marked \* perform signed comparison.

Note: the only operators which may have one or both operands relative (or external) are + and - (unary and binary). Relative + relative-relative evaluates to relative. In object module (AOF) mode, two relative operands must have the same relocation base (i.e. they must be defined as labels in the same area).

## 2.6 Mnemonic conventions

The assembler accepts all standard National Semiconductor instruction mnemonics (as described in the *Cambridge Series Instruction Set Reference Manual*), including floating point unit (FPU) and memory management unit (MMU) instructions.

The normal 32000 operand forms are accepted for all 'general' type operands, with the following conventions:

- An expression on its own is normally treated as a code-area address and is assembled as a PC-relative operand (or (SB) or EXTERNAL when the assembler is in AOF mode). The type of the expression must match the current code-area type, i.e. an absolute expression will be faulted in a relocatable area. This rule also applies to branch-type operands, i.e. of 'disp' class.
- Immediate mode operands are specified by preceding an absolute expression with the equals-sign =. In the case of floating point immediates, only a constant may follow the =, not an expression.
- Absolute operands are specified by prefixing an absolute expression with the at-sign @.

- Operands for a 'quick' type argument must be absolute expressions, optionally prefixed by an equals-sign.

In addition, the following special cases are accepted, as shown by these examples:

```
MOVSW    [U,B]
ENTER    [R0, R1, R3], 24
RESTORE  [R0-R4] ; (all registers between R0 and R4 inclusive)
SETCFG   [I]
CMPSD   []
```

## 3 Assembler directives

This chapter describes the directives acted upon by the assembler. Most of these are general purpose and may occur anywhere in the source. Others are specific to the production of Acorn Object Format files and should only be used after a `MODULE` directive.

### 3.1 Absolute and relocatable modes

At any time the assembler is in one of three modes - absolute, relocatable or AOF. The default mode is absolute. In any assembly, one (and only one) of the directives `ABSORG`, `RELOGR` or `MODULE` may occur, at most once. If one does occur, it must be before any code or data has been generated, or any label defined, otherwise it is treated as an error.

The form of these directives is:

```
ABSORG  expression
RELOGR  expression
MODULE  name
```

The value of the expression must be absolute, and defined by the time the directive is first encountered. It may not change between passes. The effect of the directive is to set the assembler into the specified mode, and the location counter to the value of the expression.

The `MODULE` directive sets AOF mode, and the optional name is planted in the output file as the module name. Once in AOF mode the assembler will allow the special directives described in section 3.3.

### 3.2 Standard directives

The following directives are handled by the assembler. As noted in the section on symbols, they may occur in either, or any combination of, upper or lower case, as may the names of user-defined macros and instruction mnemonics.