AS31 Documentation

This document is an adaptation of the original AS31 man page written by Ken Stauffer (University of Calgary), stauffer@cpsc.ucalgary.ca, who is of course the original author of AS31. All of the little changes made here at OSU are reflected in this documentation (I hope).

Usage

```
as31 [-Fformat] [-Aarg] [-l] infile.asm

as31 [-Fformat] [-Aarg] [-l] infile
```

AS31 assembles infile.asm into one of several different output formats. The output will be in a file called infile.obj. The .asm extension is required.

Command Line Options

The options must appear before the input file name. Both options are optional. The text of each flag must appear on the same argument as the flag. For example, "-Fod" is a valid argument, but "-F od" is not.

-Fformat
: This options specifies the output format that is to be used. Currently the only options available for this are:

hex
: This format is the Intel HEX format which is expected by a number of EPROM programmers and the PAULMON debugger. For use with some programmers, the output file's extension may have to be changed to .HEX to be recognized by the programmer. No -A option is used. This format should be the default if no -F option is used.

tdr
: This format generates an ascii file of hex digits formatted in such a way that they can be read by tdr's debugger. An argument can be specified (See -A option) which will pass a format specific string to the format generator. In this case, the argument string represents an offset to add to the location counter. This offset is specified in decimal and defaults to 64*1024 (0x10000). To

specify an offset of 100, you would need "-Ftdr -A100" when invoking the assembler.

byte
: This format is simply an address and a byte on each line, in ascii. No -A option is used.

od
: This format is similar to the output from od(1). The format consists of an address followed by sixteen hexadecimal bytes, followed by the ASCII equivalant. No -A option is used.

srec2, srec3, srec4
: The srecord generator is capable of generating output with any one of 2, 3, or 4 byte addresses. The -A option can be used to set the base address offset and the default here is 0x0000 (unlike tdr).

  NOTE: This assembler allows for the output formats to be expanded to include many different output formats. See the file `emitter.c` for details.

-Aarg
: This option specifies a format specific string which is passed to the format generator. Both format "tdr" and the srecord formats use this option.

-l
: This option tells the assembler to also generate a listing file. A listing will be placed in the file infile.lst. Where 'infile' is the file that is being assembled. This option may appear anywhere before infile.asm. The option must occur isolated on the command line. The listing file shows the assembler generated code in hex, and up to 60 characters are retained from the source file.

-cgi
: This option instructs the assembler to supress the listing, output the object code to stdout, and DELETE the original assembly source file. This is intended to be used with a cgi script which builds a custom configured temporary copy of an assembly source. This option should not normally be used.

Instructions

The AS31 assembler supports all of the standard 8051 family instructions. Please refer to this Instruction Set Reference sheet.

Directives

AS31 includes the following assembler directives:

.ORG expr
: Start assembling at the address specified by the expression expr. An error occurs if the assembler starts assembling over an

address space that has previously been assembled into.

.EQU symbol, expr
: Set symbol to the value of expr. The value for expr must be known during the first pass, when the line containing the .EQU is encountered.

.DB expr, expr, ...

.BYTE expr, expr, ...
: Assemble the bytes specified by the expression into memory. A string may also be specified with this directive.

.WORD expr, expr, ...
: Assemble the words specified by the expression into memory. The byte ordering used, is that used by the 8031.

.FLAG symbol1, symbol.[0-7]
: Sets symbol1 to the bit address specified by the symbol.[0-7] expression, where [0-7] denotes a character between 0 and 7. The resulting bit address is checked to see if it is a valid bit address.

.END
: This directive is ignored.

.SKIP expr
: Adds the value of expr to the location counter. Used to reserve a block of uninitialized data. Expr should be in bytes.

Language Lexical Details

All characters following a semi-colon are ignored until a newline is encountered.

All numbers default to decimal, unless the number starts with one of the following:

0x or 0X
: This indicates a hexadecimal number. ie. 0x00ff

0b or 0B
: This indicates a binary number. (1's and 0's). ie. 0b1100110010

0
: This indicates an octal number. ie. 0377

All numbers default to decimal, unless the number ends with one of the following characters:

b or B
: This indicates a binary number. Unless 0x was used above. ie. 1010101b

h or H
: This always indicates a hex number. However if the first character is not numeric, then either 0x or 0 must be specified. This avoids confusing the assembler into thinking a hex number is a symbol. For example: 0ffh, 0xffh, 0XffH, 20h, 0x20 and 020h are means to specify a valid hexdigit, but the following are not: ffh, 0ff.

d or D
: This forces a number to decimal. Unless 0X was used. ie. 129d

This causes the number to be interpreted as octal. ie. 377o

A character constant can be entered as 'c' where c is some character. \b, \n, \r, \t, \' \0 are also valid. A character constant can be used anywhere that an integer value can.

A string is entered as a set of characters enclosed in double quotes "". A string is only valid with the .BYTE directive. \b, \n, \r, \t, \" are also valid escapes. However \0 is not.

Instructions, directives, and the symbols: R0, R1, R2, R3, R4, R5, R6, R7, A, AB, and C can be entered in upper or lower case without assembler confusion. These words however cannot be defined as a user symbol. User symbols are also case insensitive.

A symbol can be any alpha numerical character plus the underscore ('_').

Expressions are accepted in most places where a value or a symbol is needed. An expression consists of the following operators. All operators evaulate to integer objects (higher precedence operators listed first):

-
: Unary minus

&
: Bit-wise AND.

|
: Bit-Wise OR.

*
: Integer multiplication.

\
: Integer division

%
: Integer modulus

+
: Integer addition.

-
: Integer subtraction.

<<
: Left bitwise shift

>>
: Right bitwise shift

In addition to these operators, a special symbol '*' may be used to represent the current location counter.

Examples

```
                .org    0
start:          mov     P3, #0xff       ; use alternate fns on P3
                                         ; leds on P1 are inverted.
                setb    F0               ; climbing up
```

```
                mov     A, #0x01        ; initial bit

write:          cpl     A               ; write it
                mov     P1, A
                cpl     A
                acall   delay
                jb      F0, climbup     ; climbing which way?

climbdn:        rr      A               ; down - shift right
                jnb     ACC.0, write    ; back for more
                setb    F0
                ajmp    write

climbup:        rl      A               ; up - shift left
                jnb     ACC.7, write    ; back for more
                clr     F0
                ajmp    write
                .end                    ; this directive ignored.
```

AS31, An Intel 8031/8051 assembler, Ken Stauffer (University of Calgary)
Minor changes and HTML markup, Paul Stoffregen
http://www.pjrc.com/tech/8051/tools/as31-doc.html
Last updated: February 24, 2005
Status: HTML version adapted from original AS31 man page
Suggestions, comments, criticisms?? <paul@pjrc.com>