AS31 Documentation

This document is an adaptation of the AS31 documentation posted on the www.prjc.com web site by Paul Stoffregen. Paul's documentation is an adaptation of the original AS31 man page written by Ken Stauffer, who is of course the original author of AS31.

Usage

```
as31 [-l] [-s] [-Fformat] [-Aarg] srcfile.asm
```

```
as31 [-l] [-s] [-Fformat] [-Aarg] srcfile
```

AS31 assembles srcfile.asm into one of several different output formats. The default output will be in a file called srcfile.hex. The .asm extension is not required on the source file or on the command line.

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.

-l

This option tells the assembler to also generate a listing file. A listing will be placed in the file srcfile.lst, where 'srcfile' is the file that is being assembled. This option may appear anywhere before srcfile.asm. The option must be 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.

-s

This option instructs the assembler to suppress the listing and send the object code to stdout. This option may appear anywhere before srcfile.asm. The option must be isolated on the command line.

-Fformat

This option specifies the output format that is to be used. This option may appear anywhere before srcfile.asm. The option must be isolated on the command line. The supported formats are:

hex

This format is the Intel HEX format which is expected by a number of EPROM programmers and the PAULMON debugger. No -A option is used. This format should be the default if no -F option is used. The output file name is srcfile.hex.

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 equivalent. 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. 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 assembler source file `emitter.c` for details.

-Aarg

This option specifies a format-specific string which is passed to the format generator. The "tdr" format and the srecord formats "srec2", "srec3", and "srec4" use this option.

Instructions

The AS31 assembler supports all of the standard 8051 family instructions. Please refer to the Instruction Set Reference included later in this document.

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, ...

Define bytes specified by the expression in memory. No space is required between values of 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. 8031 processor byte ordering is used.

.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. e.g. 0x00ff or 0xff

0b or 0B

This indicates a binary number (1's and 0's). e.g. 0b11001111

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 as described above). e.g. 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 as described above). e.g. 129d

o or O

This causes the number to be interpreted as octal. e.g. 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 ('_').

A space is optional between directive expressions and between instruction set arguments.

Labels and directives can occupy any column of the source file; they are not limited to column 1.

Expressions are accepted in most places where a value or a symbol is needed. An expression consists of the following operators. All operators evaluate 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    8000h
start:          mov     P3, #0xff       ; use alternate functions on P3
                                        ; LEDs on P1 are inverted.
                setb    F0              ; climbing up
                mov     A, #0x01        ; initial bit

write:          cpl     A               ; write it
                mov     P1,A            ; space between arguments not required
                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
                mov     P3, #0x01<<3    ; example of bit shifting
                mov     P3, #0xff/3     ; example of integer division
                ajmp    *               ; jump to current PC value
                .end                    ; this directive is ignored
```

AS31, an Intel 8031/8051 assembler, Ken Stauffer (University of Calgary), www.stauffercom.com
Minor changes and HTML markup, Paul Stoffregen, paul@pjrc.com, www.prjc.com
Minor changes and Word/PDF markup, Linden McClure, http://ece.colorado.edu/~mcclurel
http://www.pjrc.com/tech/8051/tools/as31-doc.html
Last update by Paul Stoffregen: November 28, 2003
Last update by Linden McClure: August 8, 2004
Status: Word/PDF version adapted from HTML version, adapted from original AS31 man page.

AS31: 8051 Family Instruction Set

INSTRUCTION		BYTES	CYCLES
ACALL	addr11	2	24
ADD	A, #data8	2	12
ADD	A, @Ri	1	12
ADD	A, Rn	1	12
ADD	A, direct	2	12
ADDC	A, #data8	2	12
ADDC	A, @Ri	1	12
ADDC	A, Rn	1	12
ADDC	A, direct	2	12
AJMP	addr11	2	24
ANL	A, #data8	2	12
ANL	A, @Ri	1	12
ANL	A, Rn	1	12
ANL	A, direct	2	12
ANL	C, /bit	2	24
ANL	C, !bit	2	24
ANL	C, bit	2	24
ANL	direct, #data8	3	24
ANL	direct, A	2	12
CJNE	@Ri, #data8, rel	3	24
CJNE	A, #data8, rel	3	24
CJNE	A, direct, rel	3	24
CJNE	Rn, #data8, rel	3	24
CLR	A	1	12
CLR	C	1	12
CLR	bit	2	12
CPL	A	1	12
CPL	C	1	12
CPL	bit	2	12
DA	A	1	12
DEC	@Ri	1	12
DEC	A	1	12
DEC	DPTR	1	12
DEC	Rn	1	12
DEC	direct	2	12
DIV	AB	1	48
DJNZ	Rn, rel	2	24
DJNZ	direct, rel	3	24
INC	@Ri	1	12
INC	A	1	12
INC	DPTR	1	24
INC	Rn	1	12
INC	direct	2	12
JB	bit, rel	3	24
JBC	bit, rel	3	24
JC	relative	2	24
JMP	@A + DPTR	1	24
JMP	@DPTR + A	1	24
JNB	bit, rel	3	24
JNC	relative	2	24
JNZ	relative	2	24
JZ	relative	2	24
LCALL	addr16	3	24
LJMP	addr16	3	24
MOV	@Ri, #data8	2	12
MOV	@Ri, A	1	12
MOV	@Ri, direct	2	24
MOV	A, #data8	2	12
MOV	A, @Ri	1	12

INSTRUCTION		BYTES	CYCLES
MOV	A, Rn	1	12
MOV	A, direct	2	12
MOV	C, bit	2	12
MOV	DPTR, #data16	3	24
MOV	Rn, #data8	2	12
MOV	Rn, A	1	12
MOV	Rn, direct	2	24
MOV	bit, C	2	24
MOV	direct, #data8	3	24
MOV	direct, @Ri	2	24
MOV	direct, A	2	12
MOV	direct, Rn	2	24
MOV	direct, direct	3	24
MOVC	A, @A + DPTR	1	24
MOVC	A, @A + PC	1	24
MOVC	A, @DPTR + A	1	24
MOVC	A, @PC + A	1	24
MOVX	@DPTR, A	1	12
MOVX	@Ri, A	1	24
MOVX	A, @DPTR	1	24
MOVX	A, @Ri	1	24
MUL	AB	1	48
NOP		1	12
ORL	A, #data8	2	12
ORL	A, @Ri	1	12
ORL	A, Rn	1	12
ORL	A, direct	2	12
ORL	C, /bit	2	24
ORL	C, !bit	2	24
ORL	C, bit	2	24
ORL	direct, #data8	3	24
ORL	direct, A	2	12
POP	direct	2	24
PUSH	direct	2	24
RET		1	24
RETI		1	24
RL	A	1	12
RLC	A	1	12
RR	A	1	12
RRC	A	1	12
SETB	A	1	12
SETB	bit	2	12
SJMP	relative	2	24
SUBB	A, #data8	2	12
SUBB	A, @Ri	1	12
SUBB	A, Rn	1	12
SUBB	A, direct	2	12
SWAP	A	1	12
XCH	A, #data8	2	12
XCH	A, @Ri	1	12
XCH	A, Rn	1	12
XCH	A, direct	2	12
XCHD	A, #data8	2	12
XCHD	A, @Ri	1	12
XCHD	A, Rn	1	12
XCHD	A, direct	2	12
XRL	A, #data8	2	12
XRL	A, @Ri	1	12
XRL	A, Rn	1	12
XRL	A, direct	2	12
XRL	direct, #data8	3	12
XRL	direct, A	2	12