NAME

as31 - An Intel 8031/8051 assembler

SYNOPSIS

as31 [-h] [-l] [-s] [-v] [-Aarg] [-Ffmt] [-Ofile] **infile.asm**

DESCRIPTION

As31 assembles *infile.asm* into one of several different output formats. The output will be in a file called infile.hex. The .asm extenstion is required.

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.

−h This causes the assembler to print out a verbose message describing its options. The message is written to the standard error.

−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.

−s This causes the assembler to write output to standard output.

−v This causes the assembler to print a version number to the standard output.

−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.

−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 asci file of hex digits formatted in such a way, so 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 and 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 equivilant ASCII. 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.

−Ofile This option tells the assembler to write the output to a file.

ASSEMBLER INSTRUCTIONS

This assembler accepts standard 8031/8051 instruction formats. Below is a list of instructions and addressing modes.

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
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

ASSEMBLER 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.

.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.

LEXICAL CONVENTIONS

- 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 the if the first character is non-numerical, 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

  o or O 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. Any user symbol may be used, and case will be preserved. So the user symbols "foo" and "Foo" are different, but "addc" is the same as "aDdC".

- 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 opertors listed first):

  \- Unary minus

  & Bit-wise AND.

  | Bit-Wise OR.

  * Integer multiplication.

  / Integer division

  % Integer modulus

  + Integer addition.

- Integer subtraction.

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

EXAMPLES

Below is a sample assembly program.

```
                .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.
```

AUTHORS

Ken Stauffer (University of Calgary) <stauffer@cpsc.ucalgary.ca>
Martin Langer <martin-langer@gmx.de>