

GNU Assembler (as) Command Line Essentials

(DOS prompt>) **as** (options) (A68K source file to assemble)

(options):

-a=(file name for listing file)

generate assembly listing

(file name for listing file):

Any valid file name, typically with an extension of .lis

Often the primary name is the same as the primary name of the assembler source file.

Example:

```
-a=myProg.lis
```

-I (directory path)

add (directory path) to directories search by .include directive (directory path):

Any valid path to a directory.

Example:

```
-I \68000\myIncs
```

-o (file name for object file)

name of object file generated by as

Example:

```
-o myProg.o
```

-m68000

Assemble using the M68000 processor instruction set

Example:

```
-m68000
```

--register-prefix-optional

Without this the GNU assembler expects a special character prefix on any register specification

(A68K source file to assemble):

name of file containing the M68000 source program

Complete command example:

```
C:\68000>as -a=myProg.lis -I \68000\myIncs -o myProg.o -m68000 myProg.A68
```

Some other useful DOS tips:

If you keep your program code files in a different directory than the as.exe (and ld.exe) program(s), then it is useful to set the DOS PATH to include the directory where the as.exe (and ld.exe) program files are stored. For example, if as.exe (and ld.exe) are in C:\68000 and your source program files are in

```
C:\CST8134\programs
```

then use the DOS PATH command:

```
PATH=C:\68000;%PATH%
```

to insert the C:\68000 directory before any other directories currently use by DOS when it is looking for programs.

If you create a file called A68k.bat containing:

```
as -a=%1 -I \68000\myIncs -o %1.o -m68000 --register-prefix-optional %1.A68
```

(or omit the "-I \68000\myIncs" if you don't use .include files)

then you can run the GNU assembler with much less stress by typing:

```
A68k myProg
```

(replacing "myProg" with the primary name of whatever .A68 source file you wish to assemble).

Open a separate DOS Command Window for assembling your program, from the window in which you edit the source program. This permits you to correct errors in the source edit window based on error messages displayed in the assembly window more easily.

Use a text editor for your source program which includes a display of the line number of the line where the cursor is currently located. This permits matching line numbers from error messages with source line numbers.

GNU Linker (ld) Command Line Essentials

(DOS prompt>) **ld** (options) (A68K object file(s) to link)

Sample Command:

```
ld -o %1.hex --ofORMAT=srec --cREF -Map %1.map -Ttext 1000 -Tdata 2000 -e %1 %1.o ..\lib\bdblib.o
```

(options):

-o=(file name of generated linked program)

--ofORMAT=srec

format of the generated linked program; "srec" type
required for the BDB emulator/debugger

--cREF

generate a cross-reference listing

-Map (map file name)

"(map file name)" is the file name to contain the cross-reference listing;
without this option, the cross-reference listing (if requested) is displayed
on standard output.

-Ttext nnnn

"nnnn" is a hex value which indicates the beginning address to be used for the "text" section
(use value between 0x1000 and 0x5FFF for code to be loaded into the BDB simulator).

-Tdata nnnn

"nnnn" is a hex value which indicates the beginning address to be used for the "data" section
(use value between 0x1000 and 0x5FFF for code to be loaded into the BDB simulator).

The "text" and the "code" sections must not use overlapping addresses.

-e label

"label" is the entry point (first instruction to be executed) in the program; it must have been
declared as global. The value specified for "label" is case sensitive; it must be coded here in
the same form as it was coded in the original source program.

-T (link control file)

may be used to specify a link control / script file (usually with an extension of ".lcf")
containing link commands, typically replacing many of the command line parameters
(including the "-Ttext", "-Tdata", "-e" parameters) and the object file(s) list.

-T xxxxx.lcf is particularly convenient when working with programs composed of
separately assembled object modules.

(A68K object file(s) to link):

One or more object files; usually with an extension of ".o".

Typically the first such file listed is the main program object
file and subsequent files are subroutine object files (often
from a "lib" directory of subroutine object files).

GNU **as** DIRECTIVES

SECTIONS

- .TEXT** subsection_number
default subsection_number: 0
program area containing non-modifiable values:
 - constant data (typically in subsection 1)
 - instructions (typically in subsection 0)

- .DATA** subsection_number
default subsection_number: 0
program area containing variable values

ASSEMBLY LISTING EFFECTS

- .PSIZE** lines_per_page,columns_per_line
width and depth of "printed" page for listing

- .TITLE** "page heading"
heading to be displayed at the top of each page

- .NOLIST**
do not print subsequent lines (until .LIST or end of program)

- .LIST**
print subsequent lines (until .NOLIST or end of program)

- .EJECT**
force a "new page" in listing at this point

DATA DEFINITION (usually preceded by label:)

- .ASCII** string_literal_list
ASCII encoded characters

- .ASCIZ** string_literal_list
- .STRING** string_literal_list
same as .ASCII except each string has a terminating zero value

- .BYTE** value_list
one or more byte values

- .INT** number_list
- .LONG** number_list
32-bit signed (2's complement)

.WORD number_list
.SHORT number_list
16-bit signed (2's complement)

.SKIP number_of_bytes,fill_value
.SPACE number_of_bytes,fill_value
allow space for "number_of_bytes" bytes each set to
"fill_value" (default "fill_value": 0)

.FILL repeat_count,size,value

.ALIGN number
allow zero or more bytes of memory as "slack space"
until address of next item ends in at least "number" zero valued bits

MISCELLANEOUS

.INCLUDE "path_to_source_code_file"

.EQU symbol,expression
.SET symbol,expression
symbol = expression
define a named constant

Basic Instruction Set

<size> := B | W | L (byte | word | longword)
(B:byte not valid if source or destination is An)

Copy Instructions

MOVE. <size> <source>, <destination>

flags: N,Z modified; V,C reset

permitted (basic) address mode combinations

| <u>destination:</u> | #immed | immed_addr | Dn | An | (An) |
|---------------------|--------|------------|----|----|------|
| <u>source</u> | | | | | |
| #immed | - | Y | Y | - | Y |
| immed_addr | - | Y | Y | - | Y |
| Dn | - | Y | Y | - | Y |
| An | - | Y | Y | - | Y |
| (An) | - | Y | Y | - | Y |

MOVEA. <size> <source>, An

flags: not effected

<source> may be any basic address mode;

however, LEA <source>,An is preferred for #immed to prevent possible "sign-extension" to the high-order word of An

(note LEA can not be used if <source> involves an memory access).

LEA. <size> <source>, An

flags: not effected

<source> should be #immed, Dn, or An

(see notes on MOVEA).

Transform Instructions

Arithmetic Instructions

ADD. <size> <source>, <destination>

SUB. <size> <source>, <destination>

CMP. <size> <source>, <destination>

flags: X,N,Z,V,C modified

permitted (basic) address mode combinations

| <u>destination:</u> | #immed | immed_addr | Dn | An | (An) |
|---------------------|--------|------------|----|----|------|
| <u>source</u> | | | | | |
| #immed | - | Y | Y | - | Y |
| immed_addr | - | - | Y | - | - |
| Dn | - | Y | Y | - | Y |
| An | - | Y | Y | - | Y |
| (An) | - | - | Y | - | - |

ADDA. <size> <source>, <destination>

SUBA. <size> <source>, <destination>

flags: not effected (for ADDA / SUBA)

CMPA. <size> <source>, <destination>

flags: X,N,Z,V,C modified

same as corresponding ADD, SUB, and CMP except <destination> is An

ADDQ. <size> <source>, <destination>

SUBQ. <size> <source>, <destination>

flags: X,N,Z,V,C modified

fast versions of ADD and SUB except <source> must be #immed in range #1 to #8 (inclusive)

CLR. <size> <destination>

flags: Z set; N,V,C reset

NEG. <size> <destination>

flags: X,N,Z,V,C modified

<destination> may be Dn or memory: immed_addr or (An)

Flow Control Instructions

Unconditional

BRA. <size> <immed_addr>
<size> must be B or W
<immed_addr> must be within
 -128 : +127 of PC for size B
 -32768 : +32767 of PC for size W

JMP <addr>
JSR <addr>
 <addr> may be immed_addr or (An)

RTS

Conditional

B<cond>. <size> <immed_addr>
<size> must be B or W
<immed_addr> must be within
 -128 : +127 of PC for size B
 -32768 : +32767 of PC for size W

<cond> may be

- EQ** : equal (Zero flag set)
- NE** : not equal (Zero flag clear/reset)

relevant to unsigned

- CC** : carry clear (reset)
- CS** : carry set
- HI** : higher (both Carry and Zero clear)
- LS** : lower or same (either Carry or Zero set)

relevant to 2's complement

- MI** : minus (N set)
- PL** : plus (N clear)
- VC** : overflow clear (reset)
- VS** : overflow set
- GE** : greater than or equal to
- GT** : greater than
- LE** : less than or equal to
- LT** : less than