ALD
Assembly Language Debugger 0.1.7

To run type ald

help

```
Commands may be abbreviated.
If a blank command is entered, the last command is repeated.
Type `help <command>' for more specific information on <command>.

General commands
attach         continue       detach         disassemble    display
enter          examine        file           help           ldisplay
load           next           quit           register       run
set            step           undisplay      unload

Breakpoint related commands
break          dbreak         disable        enable         ignore
lbreak         tbreak
```

General commands

help **attach**

```
attach: Attach to a running process
Usage: attach <pid>

<pid> - process id to attach to
```

help **enter**

```
enter: Change the contents of the program's memory
Usage: enter <address> [value]

<address> - Memory address to change
[value]   - New value

  If no value is given, you will be prompted for values for
successive memory addresses until a blank value is input.

Alias: store
```

help **load**

```
load: Loads a new file into memory for debugging
Usage: load <filename>

 Previous file, if any, is unloaded first
```

help **set**

```
set: Configure various settings
Usage: set [option] [value]

Options:

  args
  disasm-show-syms
  entry-point
  file-offset
  output
  pause-print
  prompt
  step-display-regs
  step-display-fpregs
  step-display-mmxregs

Type "help set <option>" for more information on <option>
```

help **continue**

```
continue: Continue execution of debugged process
Usage: continue

Alias: c
```

help **examine**

```
examine: Examine the contents of the program's memory
Usage: examine [start]|[register]|[section]|[symbol] [stop] [-num
<num>] [-size <value>] [-output <letter>]

[start]            - Memory address to start from
[register]         - Memory dump begins at register contents
[section]          - Memory dump begins at section start
[symbol]           - Memory dump begins at symbol start
[stop]             - Memory address to stop dump
[-num <num>]       - Number of elements to dump (default: 20)
[-size <value>]    - Size of each element in bytes (default: 1)
[-output <letter>] - Output format for each element (default: x)
  'x' = hexadecimal
  'o' = octal
  'd' = decimal

Example:
  examine -n 50 -s 1 -o x 0xABCD
   Dumps 50 elements each of size 1 byte in hexadecimal format,
   starting at location 0xABCD.

  If no starting address is given, the address specified in
"set entry-point" is used. A register name, section name,
or symbol name may be given in place of a starting address.

Aliases: e, dump




[start]             - Memory address to start from
[register]          - Memory dump begins at register contents
[section]           - Memory dump begins at section start
[symbol]            - Memory dump begins at symbol start
[stop]              - Memory address to stop dump
[-num <num>]        - Number of elements to dump (default: 20)
[-size <value>]     - Size of each element in bytes (default: 1)
[-output <letter>] - Output format for each element (default: x)
  'x' = hexadecimal
  'o' = octal
  'd' = decimal

Example:
  examine -n 50 -s 1 -o x 0xABCD
   Dumps 50 elements each of size 1 byte in hexadecimal format,
   starting at location 0xABCD.

  If no starting address is given, the address specified in
"set entry-point" is used. A register name, section name,
or symbol name may be given in place of a starting address.

Aliases: e, dump

ald>
help next

next: Step one instruction, stepping over any subroutines
Usage: next [num]

[num] - number of instructions to step over (default: 1)

Alias: n

ald> help step

step: Step one instruction, stepping into any subroutines
Usage: step [num]

[num] - number of instructions to step through (default: 1)

Alias: s

ald> help detach

detach: Detach from current process
```

```
Usage: detach

  Detaches the debugger from the current process (see help attach)

ald> help file

file: Outputs specified information on current file
Usage: file <header | secinfo | syminfo>

header         - Output information about the file's object header
secinfo [name] - Output information about the file's sections. If
                 [name] is given, output information about that
                 specific section.
syminfo [sym]  - Output information about the file's symbols, if any.
                 If [sym] is given, output information about that
                 specific symbol.

ald> help quit

quit: Exit the debugger
Usage: quit

ald> help undisplay

undisplay: Remove a display address
Usage: undisplay <number | all>

  number - number (can be obtained from "ldisplay")
  all    - Delete all display addresses

See also: display, ldisplay

ald> help disassemble

disassemble: Disassembles machine code into assembly language
instructions
Usage: disassemble [start [stop]] [-num <number>] [flags]

[start [stop]] - Starting and stopping memory locations - All opcodes
                 inside this range will be disassembled. For this to
                 work, you must be working with an executable file.
[-num <num>]   - Number of instructions to disassemble (default: all)
[flags]        - Various flags

Flags:
  -section <name> - disassemble specific section <name> - you can
                    use the "file secinfo" command to get a list
                    of available sections.

The output of this command is as follows:
<offset> <opcode> <instruction>
```

```
<offset>      - Virtual offset from beginning of file, or memory
address
<opcode>      - Machine language instruction
<instruction> - Assembly language instruction

 Disassembly begins at the address specified by "set file-offset",
unless a start/stop memory address is given.

Alias: d
```

ald> help **help**

```
help: Displays commands, or gives specific help on commands
Usage: help [optional commands]
```

ald> help **register**

```
register: Display and/or manipulate the process' registers
Usage: register [-all] [name [value]]

[-all]    - display all registers
[name]    - name of a specific register
[[value]] - if a name is given, it is set to this value

With no arguments, the most common registers are displayed
along with their values.
```

ald> help **unload**

```
unload: Unloads the current debug file from memory
Usage: unload
```

ald> help **display**

```
display: Display memory after single steps
Usage: display [start]|[register]|[section]|[symbol] [stop] [-num
<num>] [-size <value>] [-output <letter>]

[start]             - Memory address to start from
[register]          - Memory dump begins at register contents
[section]           - Memory dump begins at section start
[symbol]            - Memory dump begins at symbol start
[stop]              - Memory address to stop dump
[-num <num>]        - Number of elements to dump (default: 20)
[-size <value>]     - Size of each element in bytes (default: 1)
[-output <letter>] - Output format for each element (default: x)
  'x' = hexadecimal
  'o' = octal
  'd' = decimal

Example:
  display -n 50 -s 1 -o x 0xABCD
```

```
   After each single step, 50 bytes of memory starting at location
   0xABCD will be printed.

See also: ldisplay, undisplay
```

ald> help **ldisplay**

```
ldisplay: Print list of memory addresses to be displayed after single
stepping
Usage: ldisplay

See also: display, undisplay
```

ald> help **run**

```
run: Start program from beginning
Usage: run [arguments]

[arguments] - runtime arguments to pass to program - if not supplied,
              the arguments given with "set args" are used.

Alias: r
```

<u>Breakpoint related commands</u>

help **break**

```
break: Set a breakpoint
Usage: break <address | symbol>

  <address> - This is the break address. It must be set at the first
              byte of the instruction where you wish to break. You
              can use the "disassemble" command to determine
              where a specific instruction begins.
  <symbol> -  Alternatively, you can specify a debugging symbol
              such as the name of a function. The executable must
              have been compiled with debugging symbols enabled.
```

ald> help **lbreak**

```
lbreak: List all breakpoints
Usage: lbreak
```

ald> help **dbreak**

```
dbreak: Delete a breakpoint
Usage: dbreak <number | all>

  number - Breakpoint number (can be obtained from "lbreak")
  all    - Delete all breakpoints
```

```
Alias: delete

ald> help tbreak

tbreak: Set a temporary breakpoint
Usage: tbreak <address>

  <address> - Breakpoint address

 A temporary breakpoint is cleared after the first time it is hit.

ald> help disable

disable: Disable a breakpoint
Usage: disable <number | all>

  number - Breakpoint number (can be obtained from "lbreak")
  all    - Disable all breakpoints

 When a breakpoint is disabled, it has no effect until it is
reactivated using the "enable" command.

ald> help enable

enable: Reenable a breakpoint
Usage: enable <number | all>

  number - Breakpoint number (can be obtained from "lbreak")
  all    - Enable all breakpoints

 This reverses the effect of the "disable" command.

ald> help ignore

ignore: Set the ignore count for a breakpoint
Usage: ignore <number> <count>

  number - Breakpoint number (can be obtained from "lbreak")
  count  - New ignore count

 When a breakpoint has an ignore count set, it will not be
triggered until it has been hit <count> times.

ald>
```