Strength-Letter

Ralph Schmidt

COLLABORATORS			
	TITLE : Strength-Letter		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Ralph Schmidt	September 19, 2022	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

Strength-Letter

1.1 Sample AmigaGuide document

Barfly Introduction

o Assembler BASM

Supports 68000..68040 and 68881/2

VERY fast Source Passing.

Arexx-Support

Enhanced Hunk and Section Support

Pre-Compiled Includes

Memory Resident Includes

Advanced Macro Support Functions

High-Level Macros

Enhanced SAS Smalldata Format

Advanced Peephole Optimizer

Multipass Optimizer

o Debugger BDEBUG see Picture .

BDebug is an OS-Debugger that is based mainly on the Amiga Task
concept and not on the Supervisor Mode. The Debugger can open
unlimited debug sessions and the user is able to control these
session simultaneous.

The window amount is unlimited and unfortunately BDebug is very
configurable .

The SAS D1 Debug Hunk, a privat BASM Debug Hunk format and the
usual Symbol Hunk is supported.

Debug Methods:

Debug File

Debug Task

Catch Next Task
Catch Crashed Task
Special: Catch Enforcer Hits
ToolWindows
RegWindow(1 for one session)
FPUWindow(1 for one session)
DissWindow(unlimited)
MemWindow(unlimited)
CoppWindow(unlimited)
SourceWindow(unlimited)
StructWindow(unlimited)
BreakWindow(1 for one session)
WatchWindow(1 for one session)
SnoopWindow(1 for one session)
Unlimited Information Window
HunkWindow
SymbolWindow
LibraryWindow
DeviceWindow
ResourceWindow
PortWindow
ResidentWindow
InterruptWindow
Information Requesters(Task can be selected)
Taskinfo
ProcessInfo
SystemInfo
Warm,Freeze and Kill Tasks
Trace Modes
Step 1
Step X
Step DEBUG Line
Trace DEBUG Line
Trace over Calls
Trace X over Calls
Trace over OS-Calls
Trace on Flows
Trace on Address
Trace out of OS
Register Undo and History Window
A lot more functions not listed here...

1.2 arexx

If BASM is started with the option -A, the Assembler
only accepts Arexx commands.
BASM {[-Option]} File
With option -Z<Address> a Buffer can be specified,so
the assembler is able to use the Editor buffer directly.
BEND
BGETERROR - Get error
BNEXTERROR - Jmp to next error
BINITERROR - Init error list to first entry

1.3 advancedmacros

/*upper(String)
/*lower(String)
/*valof(Symbol)
/*strlen(Symbol)
/*right(String,n)
/*left(String,n)
/*mid(String,n,m)
and the normal joker:
@
\0-\99
?0
?1-?100
\#

1.4 highlevelmacros

.REG d7
.BRANCH w
.FOR d0.w = #1 to #10 STEP #2
.NEXT
.IF (a0) + #100 <> #10
.ELSE
.ENDIF
.IF d0 = #10
.ELSE

```
.ENDIF
.WHILE d0 <> #0
.ENDWHILE
.CALL func , test + #100 - #20 , #test
.RETURN d1 + d2 + #$100
.RETURN (a0)
.DEF rout , d0.w , d1 , (a0)
.ENDDEF
.LET + #100 - #10 << #7
.LET d1 = (a1) - (a0)
xref func
test:
.REG d7
.BRANCH w
.FOR d0.w = #1 to #10 STEP #2
move.w #1,d0
__for1:
addq.w #1,d1
.NEXT
add.w #2,d0
cmp.w #10,d0
blt.w __for1
.IF (a0) + #100 <> #10
move.l (a0),d7
add.l #100,d7
cmp.l #10,d7
beq.w __else1
moveq #0,d0
bra.w __endif1
.ELSE
__else1:
moveq #1,d0
.ENDIF
__endif1:
.IF d0 = #10
cmp.l #10,d0
bne.w __else2
moveq #0,d0
bra.w __endif2
```

```
.ELSE
__else2:
moveq #1,d0
.ENDIF
__endif2:
.WHILE d0 <> #0
__while1:
cmp.l #0,d0
beq.w __endwhile1
addq.w #1,d1
.ENDWHILE
bra.w __while1
__endwhile1:
.CALL func , test + #100 - #20 , #test
move.l test,d7
add.l #100,d7
sub.l #20,d7
move.l d7,-(a7)
move.l #test,-(a7)
jsr func
ifnc "8","0"
lea.l __CALLSize(a7),a7
endc
.RETURN d1 + d2 + #$100
move.l d1,d7
add.l d2,d7
add.l #$100,d7
.RETURN (a0)
move.l (a0),d7
.DEF rout , d0.w , d1 , (a0)
XDEF rout
link a5,#0
move.w $0a(a5),d0
move.l $0c(a5),d1
move.l $10(a5),(a0)
.ENDDEF
unlk a5
rts
.LET + #100 - #10 << #7
```

add.l #100,d7

sub.l #10,d7

lsl.l #7,d7

.LET d1 = (a1) - (a0)

move.l (a1),d7

sub.l (a0),d7

move.l d7,d1

1.5 smalldata

BAsm supports the SmallData specifiers

EXT_DEXT32 for 68020-40 32Bit Addressmodes

(x.l,...)

EXT_DEXT16 for the usual SAS D1 16Bit Adressmode

(x.w,...)

EXT_DEXT8 for 8Bit Adressmodes

(x.b,...)

The Smalldata register, usually A4, can be changed.

1.6 peephole

Instruction Optimizing

cmp.? #0,<ea> ->tst.? <ea>

asl.? #1,dn ->add.? dn,dn

move.b #-1,<ea> ->st <ea>

mulx #x,<ea>

.

.

.

#x Optimizing

move.l #x,dn -> moveq #y,dn+neg.w dn ;$ffff0080<=x<=$ffff0001

.

.

.

PC-Relative

Long to Word

0(an) to (an)

Register TrashReg Optimizing

trashreg d0

or.l #$14,10(a0) -> moveq #$14,d0 + or.l d0,10(a0)

trashreg a0

move.l d0,test -> lea.l test(pc),a0 + move.l d0,(a0)

Example Pass: basm -v -O xoper.s

BASM «Barfly Assembler (REG.)» Copyright © Ralph Schmidt 1990-92

Assembling : xoper.s

PASS 1

PASS 2

7755 Lines have been assembled in 0.792 Seconds and 0 Minutes.

This is a quote of 586825 Lines per Minute!

The main Part contains 6605 Lines.

1 Symbol(s) declared global and 84 Symbol(s) referenced external.

0 Includes were added to the resident Include base

5 Hunk(s) have been created.

Optimizing saves 902 Bytes Code.

Hunk: 0 Name=``

Start:0 End:$224 548 Bytes Reloc:27 Mode:CODE Memory:PUBLIC

Hunk: 1 Name=`xoper'

Start:0 End:$6248 25160 Bytes Reloc:748 Mode:CODE Memory:PUBLIC

Hunk: 2 Name=`blabla'

Start:0 End:$E64 3684 Bytes Reloc:0 Mode:BSS Memory:PUBLIC

Hunk: 3 Name=`standardpacket'

Start:0 End:$7C 124 Bytes Reloc:0 Mode:BSS Memory:PUBLIC

Hunk: 4 Name=`ICONIFY'

Start:0 End:$438 1080 Bytes Reloc:27 Mode:CODE Memory:CHIP

Errors: 0 Warnings: 0

Assembling Ok! No Errors found.

Code type is position relocatable.Output file size 33416 bytes.

1.7 multipass

Example Pass: basm -v -O -OG xoper.s

See Peephole optimizing for a comparison.

BASM «Barfly Assembler (REG.)» Copyright © Ralph Schmidt 1990-92

Assembling : xoper.s

PASS 1

Optimize Pass 1...1402 Symbol Changes | 200 Width Failures...

Optimize Pass 2...1113 Symbol Changes | 22 Width Failures...

Optimize Pass 3...1033 Symbol Changes | 2 Width Failures...

Optimize Pass 4...580 Symbol Changes | 0 Width Failures...

Optimize Pass 5...0 Symbol Changes | 0 Width Failures...

7755 Lines have been assembled in 5.132 Seconds and 0 Minutes.

This is a quote of 90665 Lines per Minute!

The main Part contains 6605 Lines.

1 Symbol(s) declared global and 84 Symbol(s) referenced external.

0 Includes were added to the resident Include base

5 Hunk(s) have been created.

Optimizing saves 2242 Bytes Code. 5 Optimize Passes needed.

Hunk: 0 Name=``

Start:0 End:$224 548 Bytes Reloc:27 Mode:CODE Memory:PUBLIC

Hunk: 1 Name=`xoper'

Start:0 End:$6160 24928 Bytes Reloc:710 Mode:CODE Memory:PUBLIC

Hunk: 2 Name=`blabla'

Start:0 End:$E64 3684 Bytes Reloc:0 Mode:BSS Memory:PUBLIC

Hunk: 3 Name=`standardpacket'

Start:0 End:$7C 124 Bytes Reloc:0 Mode:BSS Memory:PUBLIC

Hunk: 4 Name=`ICONIFY'

Start:0 End:$42C 1068 Bytes Reloc:22 Mode:CODE Memory:CHIP

Errors: 0 Warnings: 0

Assembling Ok! No Errors found.

Code type is position relocatable.Output file size 33000 bytes.

1.8 asmspeed

Testsystem: A4000+LPS240SCSI+a lot tasks

Xoper.s

Noptest.s 50000*Nop+Tab

BasmOption.s with Includes from HD

BasmOption.s with packed Includes from HD

BasmOption.s with Includes from ResidentDataBase

BasmOption.s with packed Includes from ResidentDataBase

The packed Include File does only include the System Includes, so

you can explain the difference between Mainpart and Totallines.

1.5x-4x faster as other assemblers in average

1.9 xoper

o BASM -v -f xoper.s

BASM «Barfly Assembler (REG.)» Copyright © Ralph Schmidt 1990-92

Assembling : xoper.s

PASS 1

PASS 2

7755 Lines have been assembled in 0.589 Seconds and 0 Minutes.

This is a quote of 789548 Lines per Minute!

The main Part contains 6605 Lines.

1 Symbol(s) declared global and 84 Symbol(s) referenced external.

0 Includes were added to the resident Include base

5 Hunk(s) have been created.

Optimizing saves 0 Bytes Code.

Hunk: 0 Name=``

Start:0 End:$24C 588 Bytes Reloc:27 Mode:CODE Memory:PUBLIC

Hunk: 1 Name=`xoper'

Start:0 End:$63F0 25584 Bytes Reloc:748 Mode:CODE Memory:PUBLIC

Hunk: 2 Name=`blabla'

Start:0 End:$E64 3684 Bytes Reloc:0 Mode:BSS Memory:PUBLIC

Hunk: 3 Name=`standardpacket'

Start:0 End:$7C 124 Bytes Reloc:0 Mode:BSS Memory:PUBLIC

Hunk: 4 Name=`ICONIFY'

Start:0 End:$438 1080 Bytes Reloc:27 Mode:CODE Memory:CHIP

Errors: 0 Warnings: 0

Assembling Ok! No Errors found.

Code type is position relocatable.Output file size 33880 bytes.

1.10 noptest

o BASM -v -f noptest.s

BASM «Barfly Assembler (REG.)» Copyright © Ralph Schmidt 1990-92

Assembling : noptest.S

PASS 1

PASS 2

50001 Lines have been assembled in 0.323 Seconds and 0 Minutes.

This is a quote of 9272495 Lines per Minute!

The main Part contains 50001 Lines.

0 Symbol(s) declared global and 0 Symbol(s) referenced external.

0 Includes were added to the resident Include base

1 Hunk(s) have been created.

Optimizing saves 0 Bytes Code.

Hunk: 0 Name=``

Start:0 End:$186A0 100000 Bytes Reloc:0 Mode:CODE Memory:PUBLIC

Errors: 0 Warnings: 0

Assembling Ok! No Errors found.

Code type is position independent.Output file size 100036 bytes.

1.11 basmoptionhd

BASM «Barfly Assembler (REG.)» Copyright © Ralph Schmidt 1990-92

Assembling : PTest.s

PASS 1

PASS 2

27370 Lines have been assembled in 1.864 Seconds and 0 Minutes.

This is a quote of 880703 Lines per Minute!

The main Part contains 3631 Lines.

3 Symbol(s) declared global and 1 Symbol(s) referenced external.

0 Includes were added to the resident Include base

3 Hunk(s) have been created.

Optimizing saves 0 Bytes Code.

Hunk: 0 Name=``

Start:0 End:$3174 12660 Bytes Reloc:729 Mode:CODE Memory:PUBLIC

Hunk: 1 Name=``

Start:0 End:$8FC 2300 Bytes Reloc:104 Mode:CODE Memory:PUBLIC

Hunk: 2 Name=``

Start:0 End:$214 532 Bytes Reloc:0 Mode:BSS Memory:PUBLIC

Errors: 0 Warnings: 0

Assembling Ok! No Errors found.

Code type is position relocatable.Output file size 18544 bytes.

1.12 basmoptionhdpacked

BASM «Barfly Assembler (REG.)» Copyright © Ralph Schmidt 1990-92

Assembling : PTest.s

PASS 1

PASS 2

13143 Lines have been assembled in 1.263 Seconds and 0 Minutes.

This is a quote of 624295 Lines per Minute!

The main Part contains 3631 Lines.

3 Symbol(s) declared global and 1 Symbol(s) referenced external.

0 Includes were added to the resident Include base

3 Hunk(s) have been created.

Optimizing saves 0 Bytes Code.

Hunk: 0 Name=``

Start:0 End:$3174 12660 Bytes Reloc:729 Mode:CODE Memory:PUBLIC

Hunk: 1 Name=``

Start:0 End:$8FC 2300 Bytes Reloc:104 Mode:CODE Memory:PUBLIC

Hunk: 2 Name=``

Start:0 End:$214 532 Bytes Reloc:0 Mode:BSS Memory:PUBLIC

Errors: 0 Warnings: 0

Assembling Ok! No Errors found.

Code type is position relocatable.Output file size 18544 bytes.

1.13 basmoptionresident

BASM «Barfly Assembler (REG.)» Copyright © Ralph Schmidt 1990-92

Assembling : PTest.s

PASS 1

PASS 2

29466 Lines have been assembled in 1.352 Seconds and 0 Minutes.

This is a quote of 1306861 Lines per Minute!

The main Part contains 3637 Lines.

3 Symbol(s) declared global and 1 Symbol(s) referenced external.

0 Includes were added to the resident Include base

3 Hunk(s) have been created.

Optimizing saves 0 Bytes Code.

Hunk: 0 Name=``

Start:0 End:$3174 12660 Bytes Reloc:729 Mode:CODE Memory:PUBLIC

Hunk: 1 Name=``

Start:0 End:$8FC 2300 Bytes Reloc:104 Mode:CODE Memory:PUBLIC

Hunk: 2 Name=``

Start:0 End:$214 532 Bytes Reloc:0 Mode:BSS Memory:PUBLIC

Errors: 0 Warnings: 0

Assembling Ok! No Errors found.

Code type is position relocatable.Output file size 18544 bytes.

1.14 basmoptionresidentpacked

BASM «Barfly Assembler (REG.)» Copyright © Ralph Schmidt 1990-92

Assembling : PTest.s

PASS 1

PASS 2

13143 Lines have been assembled in 0.768 Seconds and 0 Minutes.

This is a quote of 1026637 Lines per Minute!

The main Part contains 3631 Lines.

3 Symbol(s) declared global and 1 Symbol(s) referenced external.

0 Includes were added to the resident Include base

3 Hunk(s) have been created.

Optimizing saves 0 Bytes Code.

Hunk: 0 Name=``

Start:0 End:$3174 12660 Bytes Reloc:729 Mode:CODE Memory:PUBLIC

Hunk: 1 Name=``

Start:0 End:$8FC 2300 Bytes Reloc:104 Mode:CODE Memory:PUBLIC

Hunk: 2 Name=``

Start:0 End:$214 532 Bytes Reloc:0 Mode:BSS Memory:PUBLIC

Errors: 0 Warnings: 0

Assembling Ok! No Errors found.

Code type is position relocatable.Output file size 18544 bytes.

1.15 hunks

Supported Hunks

HUNK_DEBUG

SAS D1 Debug Hunk

BASM Debug Hunk

SMALLDATA

HUNK_DREL8

HUNK_DREL16

HUNK_DREL32

RELOC

HUNK_RELOC32

HUNK_RELOC16

HUNK_RELOC32SHORT

[Label] section Name[,Type[,[RelocMode,Memtype]

o Name = Hunk name

o Hunk-Type

o CODE

o DATA

o BSS

o DEBUG (you’re be able to define a Custom Debug-hunk)

o Reloc-Mode

o RELOC16 The new Reloc32Short Hunk. This Hunk does use

16Bit offsets. Available since V2.04

o Reloc32 The normal Reloc32 Hunk.

o Mem-Type

o PUBLIC

o CHIP

o FAST

o ATTR=? You can specify a custom memory attribute

Available since V2.04

1.16 BDebug Configuration file

This is the Config File for Barfly Debugger

Define Window dimensions

#COMMANDWINDOW=100/100/0/0

REGWINDOW=0/0/0/0

FPUWINDOW=428/0/0/0

DISSWINDOW=0/137/398/117/PC

MEMWINDOW=396/137/225/117/A0

COPPWINDOW=280/116/260/120/NO

BREAKWINDOW=0/137/0/0

WATCHWINDOW=0/137/0/0

STRUCTWINDOW=396/253/225/160/A0

SOURCEWINDOW=0/253/398/126/PC

GLOBALVIEWWINDOW=220/137/420/120/A0

TASKSTACK=4096

This Command adds Memory regions to the CanbeDisplayed Base that aren’t

defined in the usual Memory List and the ROM.

SHOWMEM=$e80000:$f00000

SNOOPMASK=-1

SNOOPMAX=100

Define next Window dimension

DISSWINDOW=0/200/400/120/PC

```
MEMWINDOW=380/200/260/120/A1
# Opencount for some Windows
OPENFPUWINDOWS=0
OPENDISSWINDOWS=1
OPENMEMWINDOWS=1
OPENBREAKPOINTWINDOWS=0
OPENSTRUCTWINDOWS=0
OPENSOURCEWINDOWS=0
OPENBREAKWINDOWS=0
OPENWATCHWINDOWS=0
# This Command opens the debugger on a new Screen
# If you don't specify the parameters bdebug will clone your WB-Screen
#OPENSCREEN=1448,560,2,PAL:HighRes Interlace
#SCREENFONT=topaz.font/11
# Define Output Shell
#FILESHELL=GMC:0/0/640/50/DebugShell(Custom)/CLOSE/WAIT/SMART/SCREENBARFLY
FILESHELL=CNC:0/0/640/50/DebugShell(Custom)/CLOSE/WAIT/SMART/SCREENBARFLY
# Define Source Paths for SAS D1 Debug Format
SOURCEPATH=sys:prog/asm/
SOURCEPATH=sys:prog/asm/asmtest/
SOURCEPATH=sys:prog/asm/debugtest/
# Define Command Keys to create your private Configuration
# ATTENTION: AMIGA-Keys are already used by the menus!
DEFCOMMAND=$15,CTRL,"Step 1 Position"
DEFCOMMAND=$14,CTRL,"Trace over Calls"
DEFCOMMAND=$13,CTRL,"Run Program"
DEFCOMMAND=$21,CTRL,"(PC)++"
DEFCOMMAND=$37,CTRL,"Open MemWindow"
DEFCOMMAND=$22,CTRL,"Open DissWindow"
REGFLAGS=SYMBOLS|NOBIGVIEW
#REGFLAGS=AUTOVIEWREFRESH|SYMBOLS|STACKCHECK|NOBIGVIEW
#DISSFLAGS=AUTOREFRESH|SHOWLIB|SHOWEA|WORDBRANCHES|NEGOFFSETS|NEGDI|OPCODEDATA|BLANKAFTE
STRUCTFLAGS=AUTOREFRESH|ADDRESSFORMAT
DISSFLAGS=SHOWLIB|SHOWEA|BLANKAFTERJMPBRA
LOADINCLUDE
QUIETENFORCER
CLICKTOFRONT
#SCREENINFRONT
CENTERWINDOW
```

#QUIETEXCEPTION=4 No Info Requester for Illegal Exceptions

#QUIETEXCEPTION=-1 No Info Requester at all

QUIETEXCEPTION=4

Use ILLEGALS for Trace_Subroutine instead of Single Stepping

TRACEBREAK

#DISABLEXPOINTER

DONOTCACHEFULLFILE