Software packages for 32000 series

Author: Alexander Voropai alec@sensi.org
Aug 2017 .. Jun 2022
This page was created as a “companion” article for this amazing site: http://www.cpu-ns32k.net [http://www.cpu-ns32k.net]
For the GNU Toolchains for 32000 series see a separate page: ns32ktoolchain

Definicon

Definicon DSI-32 was an “coprocessor” ISA-8 board with NS32032 CPU designed to run a “heavy” match tasks.

See a Byte Magazine articles:
https://archive.org/details/BYTE_Vol_10-08_1985-08_The_Amiga [https://archive.org/details/BYTE_Vol_10-08_1985-08_The_Amiga]
p. 120 The DSI-32 Coprocessor Board. Part 1: The Hardware
https://archive.org/details/byte-magazine-1985-09 [https://archive.org/details/byte-magazine-1985-09]
p. 116 The DSI-32 Coprocessor Board. Part 2: The Software

Micro Cornucopia magazine article: http://bitsavers.trailing-edge.com/pdf/microCornucopia/Micro_Cornucopia_%2325_Aug85.pdf [http://bitsavers.trailing-edge.com/pdf/microCornucopia/Micro_Cornucopia_%2325_Aug85.pdf]
p.52 The Definicon DSI-32 Co-Processor

Interesting issue about “Scienstific coprocessors” for the early PCs (PC Mag 14 Apr 1987)
https://books.google.ru/books?id=Jr7Bj4Yr0FMC&printsec=frontcover [https://books.google.ru/books?id=Jr7Bj4Yr0FMC&printsec=frontcover]
p.208 “Changing the rules of the game”
DSI-32 and similar boards

Udo Moeller's site:
Hardware: http://cpu-ns32k.net/Defhard.html [http://cpu-ns32k.net/Defhard.html]
Software: http://cpu-ns32k.net/Defsoft.html [http://cpu-ns32k.net/Defsoft.html]

NEWS: A blog post about DSI-32 board:
https://virtuallyfun.com/category/ns32032/ [https://virtuallyfun.com/category/ns32032/]

DSI-32 Software

1) http://user.berklix.org/~dlr/dsi/set021387/ [http://user.berklix.org/~dlr/dsi/set021387/]
This is a complete Definicon software set. Includes GreenHills C (K&R only), Fortran and Pascal compilers. Filenames are in mixed case. An assembler is made by the Computer Systems Design (Perth, Western Australia) and language compilers are from the Green Hills Software. Assembler is quite powerful and supports **module** concept for the 32000 (unlike many other assemblers including `gas`). There are also two alternative assemblers (included). Object module format is proprietary. Executables has a suffix `*.e32` and this is an `a.out` variant (loader sources is available in the LOAD sources).

```
CSD-32000 Assembler V1.00a, Computer Systems Design
C-32000 1.6.12 Copyright(c)1985 Green Hills Software, Inc.
Fortran-32000 1.6.12 Copyright(c)1985 Green Hills Software, Inc.
Pascal-32000 1.6.12 Copyright(c)1985 Green Hills Software, Inc.
DSI-32000 Linker V1.03c, Definicon Systems Inc
```

Pascal and Fortran are in the ARC archives (old DOS compressor, available at Simtel CD and mirrors)

2) http://cpu-ns32k.net/files/Def-Soft.zip [http://cpu-ns32k.net/files/Def-Soft.zip]
This archive contains a more recent assembler, but not a full set. Includes 32IO and LOAD sources(!). Filenames in Uppercase. Also includes some Public Domain utilities.

```
DSI-32000 Assembler V1.03m, Definicon Systems Inc.
DSI-32000 Linker V1.03d, Definicon Systems Inc
```

Documentation: http://cpu-ns32k.net/files/docs.zip [http://cpu-ns32k.net/files/docs.zip]

This `c` compiler (was tested under emulator) is trying to include a “system” HOST header files from the `/usr/include`. This does not work because of non-K&R syntax and complexity of the modern headers. Use a “local” header files (in double quotes) `#include “stdio.h”` in your sources.
(TODO: Make a “fake root” in the emulator ?) C Compiler options: dsi32-copt

`ln` linker has unusual syntax. It requires a special “definition file” (slightly similar to the “response file” for a Microsoft's linker). Software distribution provides sample files with `def` or `d32` extension.
LN users manual: http://cpu-ns32k.net/files/LN_user_manual.pdf [http://cpu-ns32k.net/files/LN_user_manual.pdf]

Emulator

As it is clear, Definicon software requires a real DSI-32 card with a real 32000 CPU installed in a PC. Fortunately, in 2016 Dave Rand wrote a **DSI-32 emulator** using a 32016 software emulator from the Acorn 32016 Tube Coprocessor Board project:
https://github.com/hoglet67/PiTubeClient [https://github.com/hoglet67/PiTubeClient]
https://github.com/hoglet67/PiTubeDirect [https://github.com/hoglet67/PiTubeDirect]

DSI-32 emulator:
http://ftp.berklix.org/~dlr/dsi/emu32k/ [http://ftp.berklix.org/~dlr/dsi/emu32k/]

I did a slightly improved version:
- Renamed emulator executable file to “load” to conform DSI-32 name loader
- Added “file normalization” to translate ALL filenames to lowercase.
- Commented on service calls
- Many other small improvements (and new bugs)
http://wiki.sensi.org//download/ns32k/emu32k_load_10a.tgz [http://wiki.sensi.org//download/ns32k/emu32k_load_10a.tgz]

With this patch you can use lowercase filenames everywhere. File extensions is unnecessary. All options after `-a` are passed to the 32000 program (like as real LOAD.EXE).

```
./load cc -a -O -ge -X88 test.c
./load as -a test
./load ln -a file=test
./load test
```

A bash command to rename all files in the current directory to lowercase:

```
for i in *; do mv $i `echo $i | tr [:upper:] [:lower:]`; done
```

A `mon2m` is a symbolic debugger. To use a symbolic debugger you have to make a `<name>.s32` file with linker directive `symbol`. Debugger will load it automatically. Second argument `<name>.e32` is a program to debug. It becomes `argv[0]` argument. Symbolic names must start with . to inform the debugger that the string is a name and not a hexadecimal input:

```
./load ln -a file=test map symbol
./load mon2m test
*L._main
Module name = test.c, Symbol name = _main
008000  ENTER   [R3,R4,R5],H'0
008003  ADDR    H'0(SB),TOS
008009  CXP     _printf
00800B  CMPQD   H'0,TOS
00800D  ADDR    H'1C(SB),TOS
008013  CXP     _printf
008015  CMPQD   H'0,TOS
008017  ADDR    @H'7F,TOS
00801B  MOVD    H'DEAD,TOS
008021  ADDR    @H'4D2,TOS
008025  ADDR    H'2E(SB),TOS
00802B  CXP     _printf
*
```

Trace does not work since not implemented in the 32016 emulator. TODO?

Portable C Compiler

Portable C compiler [https://en.wikipedia.org/wiki/Portable_C_Compiler] (pcc) ported to 16032:
http://www.tuhs.org/Archive/Applications/Portable_CC/ [http://www.tuhs.org/Archive/Applications/Portable_CC/]
Contains **a16**, **cc16** and **ld16** sourcecode. Binary format is **a.out** variant.

Not tested.

Labtam

http://www.cpu-ns32k.net/Multi.html [http://www.cpu-ns32k.net/Multi.html]

Labtam firmware:
http://wiki.sensi.org/download/ns32k/Labtam-V32-G87.zip [http://wiki.sensi.org/download/ns32k/Labtam-V32-G87.zip]

It supports the following Multibus controllers:

```
Disk:
* Labtam Z80 disk controller *
* Interphase STORAGER disk *
* Interphase 2190 SMD Controller *
Tape:
* Labtam Streaming Tape Interface *
* XYlogics 472 Tape controller *
* Interphase STORAGER tape *\\
```

TDS

TDS (Tiny Development System) is a software for the National Semiconductor's development boards: **DB16000**, **DB16000A** (later renamed to **DB32016**) and **DB32000**. It includes simply editor, assembler and debugger.
http://cpu-ns32k.net/National.html [http://cpu-ns32k.net/National.html]
http://bitsavers.trailing-edge.com/components/national/db32016/ [http://bitsavers.trailing-edge.com/components/national/db32016/]
https://www.scss.tcd.ie/SCSSTreasuresCatalog/hardware/TCD-SCSS-T.20141120.007/TCD-SCSS-T.20141120.007.pdf [https://www.scss.tcd.ie/SCSSTreasuresCatalog/hardware/TCD-SCSS-T.20141120.007/TCD-SCSS-T.20141120.007.pdf]

DBMON Reference Manual [http://wiki.sensi.org/download/ns32k/dbmon.pdf]

Sourcecode: http://wiki.sensi.org/download/ns32k/tds.tgz [http://wiki.sensi.org/download/ns32k/tds.tgz] (may be not complete). Assembler dialect is compatible with DSI assembler and may be compiled and linked.

GENIX

GENIX was a National Semiconductor's trademark for the UNIX-class OS for the NS32000 architecture. Early GENIX was a 4.xBSD port and later GENIX was a UNIX SYSV (R2 and R3) port. A significant number of workstations used this OS: i.e. Whitechapel Computer Works [http://cpu-ns32k.net/Whitechapel.html], Trinity Workstation [http://cpu-ns32k.net/Trinity.html] and NS SYS32/20 [http://cpu-ns32k.net/Opus.html]. There were also proprietary 32000 UNIX ports under it's own names (i.e. Sequent DYNIX [https://en.wikipedia.org/wiki/DYNIX], Siemens MX 300 [http://cpu-ns32k.net/Siemens.html] SINIX [https://en.wikipedia.org/wiki/SINIX], Encore [http://cpu-ns32k.net/Encore.html] UMAX, Tektronix UTek e.t.c)

GENIX Documentation

http://bitsavers.trailing-edge.com/components/national/_dataBooks/1983_Development_Systems_Products_Databook.pdf [http://bitsavers.trailing-edge.com/components/national/_dataBooks/1983_Development_Systems_Products_Databook.pdf]
page 118 (SYS16 system, running early GENIX version based on the 4.1BSD)

http://bitsavers.trailing-edge.com/components/national/_dataBooks/1988_National_Series_32000_Microprocessors_Databook.pdf [http://bitsavers.trailing-edge.com/components/national/_dataBooks/1988_National_Series_32000_Microprocessors_Databook.pdf]
section 7-16 (GENIX/V.3 is a port of AT&T's System V, Release 3.0, UNIX operating system for the Series 32000 microprocessor family.)

Tilson, Michael (October 1983). “Moving Unix to New Machines [https://archive.org/stream/byte-magazine-1983-10/1983_10_BYTE_08-10_UNIX#page/n267/mode/2up]”. BYTE. p. 266. An article about porting early UNIX to the NS32000 architecture. Seems this article is a source of the “unreliability” opinion about NS32000 CPU.

Since GENIX 2 is based on the AT&T SYSVR2/3 hence most of documentation are suitable for the GENIX 2 too.
http://bitsavers.trailing-edge.com/pdf/att/unix/System_V_Release_2/ [http://bitsavers.trailing-edge.com/pdf/att/unix/System_V_Release_2/]
http://bitsavers.trailing-edge.com/pdf/att/unix/System_V_Release_3/ [http://bitsavers.trailing-edge.com/pdf/att/unix/System_V_Release_3/]

GENIX distributives

1) http://bitsavers.trailing-edge.com/bits/AmericanInformationSystems/AIS_GENIXsrc_6-20-85.tar.gz [http://bitsavers.trailing-edge.com/bits/AmericanInformationSystems/AIS_GENIXsrc_6-20-85.tar.gz]
This is an early GENIX sources (derived from the 4.1BSD) for the DB16000 board. Also includes SNXC (an early 32000 crosscompiler) for the VAX 4.1BSD. An assembler sourcecode is written in Pascal (sic!). C compiler is PCC. Binary format is a.out (variant). See below how to run this SNXC at a VAX emulator.

2) http://bitsavers.trailing-edge.com/bits/AmericanInformationSystems/AIS_oldUsr_10-3.cpio.gz [http://bitsavers.trailing-edge.com/bits/AmericanInformationSystems/AIS_oldUsr_10-3.cpio.gz]
This ia a /usr cpio archive from a more recent GENIX (SYSVR2). No sources, no /bin, /lib and /etc. The /usr/bin contains binaries in the ns32k COFF format (similar to the Opus5 binaries, see below). Some binaries are not stripped.

3) http://wiki.sensi.org/download/ns32k/SysVr2.0_32000.tgz [http://wiki.sensi.org/download/ns32k/SysVr2.0_32000.tgz]
This is a full SYSVR2 sources ported to the NS32000 processor for National Semiconductor SYS16/SYS32 machines. Includes NS32000 assembler, loader and C compiler sources. This UNIX sources correspond to the well known SYSVR2 sources for the VAX and 3B2 SYSVR2 floating around: i.e. sysvr2-vax.cpio [http://vetusware.com/download/UNIX%20SYSVr2%20Source%20Code%20r2%20_usr%20dump_%20r2/?id=13474] but for the NS32000 CPU.

The `as` assembler is written in a “classic” LEX/YACC. Unusually, `m4` is used as macro-pre-processor. Unfortunately, this assembler is very “fragile” - it coredumps on any errors in the input file. Object file format for this `as` and `ld` is COFF (ns32k variant). Assembler has a quite unusual syntax: all numerical constants and variables are treated as “absolute” (even without “@var”). According to some reports, this addressing mode works faster. The syntax for the `var(pc)` (PC-relative) addressing mode is unknown.

Sources may be cross-compiled under working SYSVR2 (i.e. SIMH/VAX or SIMH/3B2) to make a sort-of “crosstool”. It is possible also to build binaries under relative old GCC (I've successfully built some sources with GCC 2.7 on a i386 RedHat Linux 2.1 Bluesky [https://archive.org/details/RHL2.1], November 1995 under VirtualBox).

Opus Systems board

1) http://bitsavers.trailing-edge.com/bits/OpusSystems/Floppies/ [http://bitsavers.trailing-edge.com/bits/OpusSystems/Floppies/]
This is a full binary distributive for the Opus5 card (see http://cpu-ns32k.net/Opus.html [http://cpu-ns32k.net/Opus.html]) also know as a National Semiconductor SYS32/20 Development Board. It's SYSVR2 (equal to GENIX2 ?). No sourcecode. Binaries are in the ns32k COFF format. Contains some DOS utuluties for the Opus5 board.

2) http://bitsavers.org/pdf/opusSystems/32k/ [http://bitsavers.org/pdf/opusSystems/32k/]
Opus Systems board documentation.

Siemens SINIX

Siemens SINIX-H [https://en.wikipedia.org/wiki/SINIX] is an UNIX-class operating system for the Siemenx MX [http://cpu-ns32k.net/Siemens.html] line of the computer based on the NS32000 CPU.

http://oldcomputers.dyndns.org/public/pub/rechner/siemens/mx-rm/index.html [http://oldcomputers.dyndns.org/public/pub/rechner/siemens/mx-rm/index.html]
Floppy images for SINIX-H. All floppy disks are copied twice: in Catweasel [https://en.wikipedia.org/wiki/Individual_Computers_Catweasel] floppy backup hardware (*.DMK) and ImageFloppy tool (*.IMD). SINIX uses unusual 80-track SD/DD variable-density floppy: Track0 is in FM 16×128, while all another tracks 1-79 are in MFM 16×256. These images are bit-copy of tracks including sector's headers so you have to translate it to the raw-data format (w/o headers). Adam Sampson's dumpfloppy [https://offog.org/code/dumpfloppy/] can extract individual tracks from the *.IMD files into raw format.

First 3 diskettes are bootable disks(?). Diskettes 005..006 is a kernel+miniroot(?) image. Diskette 004 and starting from Diskette 007 contains installation data as concatenated Install+Data tar files (Data tar starts at offset `0x6000`) (Update: Diskette 013,014,020,021 contains productive kernel + root filesystem dump(?), not a tar). To extract diskettes:

```
# imdcat SINI_007.IMD -o SINI_007.RAW -c 1:79 -n
IMD 1.18: 26/01/2018 19:39:16
U 11063-C999
SINIX/LZS-HD
```

```
V5.21 / MX300-05
07.08.1989 | 3-4 | LP9022
#
# tar tvf SINI_007.RAW
-rwx--x--x 0/12           1884 1989-08-07 15:10:03 install
-rw------- 0/12             29 1989-08-03 17:57:23 install.conf
#
# dd if=SINI_007.RAW of=SINI_007_.RAW  bs=1 skip=24576
622592+0 records in
622592+0 records out
622592 bytes (623 kB) copied, 1.97945 seconds, 315 kB/s
#
# tar tvf SINI_007_.RAW
-rwx--x--x 0/1          126976 1989-08-01 11:05:02 lp9022
-rwx--x--x 0/1           24580 1989-08-03 13:23:01 MX200/dump.sa
-rwx--x--x 0/1           32772 1989-08-03 13:38:03 X20/dump.sa
-rwx--x--x 0/1           28676 1989-08-03 13:14:54 X9733/dump.sa
-rwx--x--x 0/1           28676 1989-08-03 13:14:38 X9733L/dump.sa
```

http://oldcomputers.dyndns.org/public/pub/rechner/siemens/manuals/Sinix/index.html [http://oldcomputers.dyndns.org/public/pub/rechner/siemens/manuals/Sinix/index.html]
Manuals in German language

SINIX uses `a.out` format (variant) with extensive use of ns32k architecture *module* feature. There is no known tools to examine this executebles.

SNX/GNX crosstools

National Semiconductor's SNXC16 (later known as NSX-32 and GNX (GENIX Native and Cross-Support)) was a cross-tool including C, Fortran and Pascal compilers, linker, assembler and supporting tools. Assembler accepts an “old” or “classic” NS16000/32000 syntax.

- http://bitsavers.trailing-edge.com/components/national/_dataBooks/1983_Development_Systems_Products_Databook.pdf [http://bitsavers.trailing-edge.com/components/national/_dataBooks/1983_Development_Systems_Products_Databook.pdf]
page 121 (an early XNSC16 version for VAX 4.1BSD. See below how to run it.)
- http://bitsavers.trailing-edge.com/components/national/_dataBooks/1986_National_NS32000_dataBook.pdf [http://bitsavers.trailing-edge.com/components/national/_dataBooks/1986_National_NS32000_dataBook.pdf]
section 9-7 (a newer GNX version for 4.2BSD, I have no this version)
- http://bitsavers.trailing-edge.com/components/national/_dataBooks/1988_National_Series_32000_Microprocessors_Databook.pdf [http://bitsavers.trailing-edge.com/components/national/_dataBooks/1988_National_Series_32000_Microprocessors_Databook.pdf]
section 7-3 (GNX Release 2 for VAX 4.2BSD, VAX VMS or Unix SYSVR3. I have no this version)
- http://bitsavers.trailing-edge.com/components/national/_dataBooks/1989_National_Embedded_System_Processor_Databook.pdf [http://bitsavers.trailing-edge.com/components/national/_dataBooks/1989_National_Embedded_System_Processor_Databook.pdf]
section 5-16 (GNX Version 3 for VAX BSD,VMS and Ultrix or NS Unix SYSVR3 port. I have no this version)

Distributives:
http://oldcomputers-ddns.org/public/pub/rechner/national_semiconductors_series_32000/index.html [http://oldcomputers-ddns.org/public/pub/rechner/national_semiconductors_series_32000/index.html]
GNX v3 and GNXv4 documentation and GNX v4 MS-DOS executables.

How to run VAX SNXC16

Early SNXC16 crosstools for VAX 4.1BSD is available at Bitsavers:
http://bitsavers.org/bits/AmericanInformationSystems [http://bitsavers.org/bits/AmericanInformationSystems]
AIS_GENIXsrc_6-20-85 contains a SNX16 binaries. To run this crosstools binaries you have to have a working 4.1BSD or 4.1cBSD on a real or simulated VAX (SNXC16 does not work under 4.2BSD). This archive also contains sourcecode.

```
NS16032 C compiler release 5.1 9/8/83 (JZ,jima)
NSC 16032 MESA ASSEMBLER Version 3.31    9/19/83
```

Fortunately, SIMH emulator provides nice VAX emulation. Run a SIMH VAX, create a .tap file and “import” SNXC16 into working 4.1cBSD
http://gunkies.org/wiki/Installing_4.1_BSD_on_SIMH [http://gunkies.org/wiki/Installing_4.1_BSD_on_SIMH]
http://gunkies.org/wiki/Installing_4.1c_BSD_on_SIMH [http://gunkies.org/wiki/Installing_4.1c_BSD_on_SIMH]

Network was in the “embrionic” stage in the 4.1BSD. An emulated “tape” may be used to “import” and “export” between emulated OS and host. To extract a .tap SIMH tape image there is a “extracters/rawtap” ulility from the SIMH's utils: https://github.com/simh/simtools [https://github.com/simh/simtools]

SNXC16 man pages extracted from this archive: ar.1 [http://wiki.sensi.org/download/ns32k/ar_1.pdf] as.1 [http://wiki.sensi.org/download/ns32k/as_1.pdf] cc.1 [http://wiki.sensi.org/download/ns32k/cc_1.pdf] ddt.1 [http://wiki.sensi.org/download/ns32k/ddt_1.pdf] ld.1 [http://wiki.sensi.org/download/ns32k/ld_1.pdf] monitor.3 [http://wiki.sensi.org/download/ns32k/monitor_3.pdf] nm.1 [http://wiki.sensi.org/download/ns32k/nm_1.pdf] ranlib.1 [http://wiki.sensi.org/download/ns32k/ranlib_1.pdf]

Output binary file is a.out.5 [http://wiki.sensi.org/download/ns32k/a_out_5.pdf]

NSXC16 Assembler Reference Manual [http://wiki.sensi.org/download/ns32k/as_refman.pdf]

How to run MS-DOS GNX 4

There were a GNX v3 and GNX v4, a cross tools for the National Semiconductor's evaluation boards: NSV-CG16-EDB, NSV-CG160-EDB, NSV-GX32-EDB, NSV-GX320-EDB, NSV-FX-GC-EDB, GC160LX-LBP, NSP-AM160-EDB, FX16FAX

There is very few information about these boards. I've found this article: https://archive.org/details/the-computer-journal-43 [https://archive.org/details/the-computer-journal-43] page 33

It is still possible to run a GNX tools under DosBox [https://www.dosbox.com/] even on the recent Windows system (including a modern 64-bit Windows 10). DosBox is an emulator program which emulates an IBM PC compatible computer running a DOS operating system.

Download a GNX 4.4 http://oldcomputers-ddns.org/public/pub/rechner/national_semiconductors_series_32000/gnx/nsc-gnx-4.4_images.rar [http://oldcomputers-ddns.org/public/pub/rechner/national_semiconductors_series_32000/gnx/nsc-gnx-4.4_images.rar]

Add the following options to the `dosbox-0.74.conf` (C:\Users\%USERNAME\AppData\Local\DOSBox) :

```
...
...
[autoexec]
# Lines in this section will be run at startup.
# You can put your MOUNT lines here.
@echo off
mount c c:\TEMP
set GNXDIR=C:\GNX
set PATH=%PATH%;C:\GNX
C:
```

Extract GNX 4.4 folder (Software\folder.gnx) into host i.e. C:\TEMP\GNX (programs will see it as C:\GNX inside DosBox)

DosBox provides 16MB of DOS XMS memory by default (GNX/MS-DOS uses built-in DOS Extender).

GNX v3 and v4 use relative modern `COFF` format (with debug info, modules support e.t.c). GNX4 cross-tools has a SYS5_2 and SYS5_3 (with shared libraries support) targets. For more info read the “GNX Language tools 4” book “COFF Programmers Guide” chapter.

GNX v4 C compiler supports ANSI C and function prototypes.