КОНСТРУИРОВАНИЕ ПРОГРАММ

Лекция № 04.1 Ассемблер NASM

+375 17 293 8039 (505a-5)

+375 17 320 7402 (ОИПИ НАНБ)

prep@lsi.bas-net.by

ftp://student:2ok*uK2@Rwox@lsi.bas-net.by/

Кафедра ЭВМ, 2022

2022.03.04

Оглавление

Язык ассемблера NASM

Основная литература и источники информации

1) NASM – The Netwide Assembler (**/usr/share/doc/nasm-doc/nasmdoc.pdf**)

2) Расширенный ассемблер: NASM (**http://www.opennet.ru/docs/RUS/nasm/**)

3) Столяров А. В. Программирование: введение в профессию. II: Низкоуровневое программирование. — М.: МАКС Пресс, 2016. — 496 с.

4) Compiler Explorer (**https://godbolt.org/**)

Установка из дистрибутива

Хотя все необходимое присутствует в составе любого дистрибутива Linux

```
$ dnf list | grep nasm
nasm.x86_64                2.15.05-1.fc34    updates
nasm-doc.noarch            2.15.05-1.fc34    updates
nasm-rdoff.x86_64          2.15.05-1.fc34    updates
...
vim-syntastic-nasm.noarch  2.15.05-1.fc34    updates
$ dnf install -y nasm nasm-doc
...
```

имеет смысл собрать последнюю версию ассемблера самостоятельно.

Веб сайт NASM: **https://www.nasm.us/**. (Стабильная версия на 01.03.2022 — **2.15.05**)

Самостоятельная сборка nasm и установка

1) Создаем каталог, куда скачаем дистрибутив nasm и где будем его собирать, и сделаем его рабочим

```
$ mkdir -p ~/software/nasm && cd ~/software/nasm
```

2) скачиваем с **https://www.nasm.us/** последнюю стабильную версию.
На сегодня это 2.15.05 от 2020.08.28.

```
$ wget -c https://www.nasm.us/pub/nasm/releasebuilds/2.15.05/nasm-2.15.05.tar.xz
```

Если wget ругается на просроченный сертификат, используем опцию **--no-check-certificate**

3) Распакуем архив и подготовимся к сборке

```
$ tar -xf nasm-2.15.05.tar.xz
$ ll
итого 1788
drwxrwxr-x. 24 podenok podenok   4096 авг 28 19:04 nasm-2.15.05
-rw-rw-r--.  1 podenok podenok 995732 авг 28 19:04 nasm-2.15.05.tar.xz
-rw-rw-r--.  1 podenok podenok 823612 авг 28 19:05 nasm-2.15.05-xdoc.tar.xz
$ cd nasm-2.15.05
```

4) Читаем файл **INSTALL** и конфигурируем сборку

```
$ ./configure --help # посмотрим ради любопытства, что можем поменять
$ ./configure --docdir=/usr/local/share/doc/nasm   # конфигурирование
checking for prefix by checking for nasm... /usr/local/bin/nasm
...
checking if gcc supports C99 external inlines... yes
checking if gcc supports typeof... __typeof
configure: creating ./config.status
config.status: creating Makefile
config.status: creating doc/Makefile
config.status: creating config/config.h
config.status: config/config.h is unchanged
```

В результате будет сгенерирован файл управления сборкой — **Makefile**.

Полезно сначала запустить конфигурирование с выводом сообщений в **/dev/null**, чтобы увидеть чего не хватает для полной сборки

```
$ make distclean                  # на всякий случай очищаем каталог то лишних файлов
$ ./configure --docdir=/usr/local/share/doc/nasm > /dev/null
configure: WARNING: No asciidoc package found, cannot build man pages
configure: WARNING: No xmlto package found, cannot build man pages
$ # устанавливаем не найденное и повторно конфигурируем, пока выход не станет чистым
$ ./configure --docdir=/usr/local/share/doc/nasm > .nasm.configure.log
$
```

5) Проверяем наличие необходимых шрифтов для сборки документации, а именно **SourceSansPro** (нет кириллицы, зато человеческая ell), **ClearSans**, **Helvetica**, **Arial**. Для этого запускаем сборку с документацией

```
$ make -j8 everything > .nasm.make.log
```

Если сборка завершается с сообщением о недоступности вышеупомянутых шрифтов, их следует установить и повторить сборку. После чего следует установка (в **/usr/local/**):

```
$ sudo make install_everything
$ nasm --version
$ NASM version 2.15.05 compiled on Feb 23 2021
$
```

Сходим на место установки и посмотрим, что там есть

```
$ cd /usr/local
$ find -type f -mtime 0
./bin/nasm
...
./share/doc/nasm/nasmdoc0.html
./share/doc/nasm/nasmdoc.txt
./share/doc/nasm/nasmdoc.pdf
./share/man/man1/nasm.1
...
```

Запуск NASM

Для ассемблирования файла необходимо ввести команду:

```
$ nasm -f <format> <filename> [-o <output>]
```

Например, команда

```
$ nasm -f elf myfile.asm
```

будет ассемблировать **myfile.asm** в ELF-объектный файл **myfile.o**.

```
$ nasm -f bin myfile.asm -o myfile.bin
```

будет ассемблировать **myfile.asm** в обычный бинарный файл **myfile.bin**.

Для получения файла-листинга, содержащего слева от оригинального исходного текста шестнадцатиричные коды, генерируемые NASM, следует использовать опцию **-l:**

```
$ nasm -f coff myfile.asm -l myfile.lst
```

Получение информации о версии NASM:

```
$ nasm -v
NASM version 2.14.02 compiled on Mar  5 2020
```

Получение справки по командной строке NASM:

```
$ nasm -h
Usage: nasm [-@ response_file] [options...] [--] filename
       nasm -v (or --v)

Options (values in brackets indicate defaults):

    -h            show this text and exit (also --help)
    -v (or --v)   print the NASM version number and exit
    -@ file       response file; one command line option per line

    -o outfile    write output to outfile
    --keep-all    output files will not be removed even if an error happens

    -Xformat      specifiy error reporting format (gnu or vc)
    -s            redirect error messages to stdout
    -Zfile        redirect error messages to file

    -M            generate Makefile dependencies on stdout
    -MG           d:o, missing files assumed generated
    -MF file      set Makefile dependency file
    -MD file      assemble and generate dependencies
    -MT file      dependency target name
    -MQ file      dependency target name (quoted)
    -MP           emit phony targets

    -f format     select output file format
       bin                  Flat raw binary (MS-DOS, embedded, ...) [default]
       ith                  Intel Hex encoded flat binary
```

```
   srec                    Motorola S-records encoded flat binary
   aout                    Linux a.out
   aoutb                   NetBSD/FreeBSD a.out
   coff                    COFF (i386) (DJGPP, some Unix variants)
   elf32                   ELF32 (i386) (Linux, most Unix variants)
   elf64                   ELF64 (x86-64) (Linux, most Unix variants)
   elfx32                  ELFx32 (ELF32 for x86-64) (Linux)
   as86                    as86 (bin86/dev86 toolchain)
   obj                     Intel/Microsoft OMF (MS-DOS, OS/2, Win16)
   win32                   Microsoft extended COFF for Win32 (i386)
   win64                   Microsoft extended COFF for Win64 (x86-64)
   ieee                    IEEE-695 (LADsoft variant) object file format
   macho32                 Mach-O i386 (Mach, including MacOS X and variants)
   macho64                 Mach-O x86-64 (Mach, including MacOS X and variants)
   dbg                     Trace of all info passed to output stage
   elf                     Legacy alias for "elf32"
   macho                   Legacy alias for "macho32"
   win                     Legacy alias for "win32"


-g              generate debugging information
-F format       select a debugging format (output format dependent)
-gformat        same as -g -F format
   elf32:       dwarf       ELF32 (i386) dwarf (newer) [default]
                stabs       ELF32 (i386) stabs (older)
   elf64:       dwarf       ELF64 (x86-64) dwarf (newer) [default]
                stabs       ELF64 (x86-64) stabs (older)
   elfx32:      dwarf       ELFx32 (x86-64) dwarf (newer) [default]
                stabs       ELFx32 (x86-64) stabs (older)
   obj:         borland     Borland Debug Records [default]
```

```
   win32:     cv8       Codeview 8+ [default]
   win64:     cv8       Codeview 8+ [default]
   ieee:      ladsoft   LADsoft Debug Records [default]
   macho32:   dwarf     Mach-O i386 dwarf for Darwin/MacOS [default]
   macho64:   dwarf     Mach-O x86-64 dwarf for Darwin/MacOS [default]
   dbg:       debug     Trace of all info passed to debug stage [default]

-l listfile   write listing to a list file
-Lflags...    add optional information to the list file
   -Lb        show builtin macro packages (standard and %use)
   -Ld        show byte and repeat counts in decimal, not hex
   -Le        show the preprocessed output
   -Lf        ignore .nolist (force output)
   -Lm        show multi-line macro calls with expanded parmeters
   -Lp        output a list file every pass, in case of errors
   -Ls        show all single-line macro definitions
   -Lw        flush the output after every line (very slow!)
   -L+        enable all listing options except -Lw (very verbose!)

-Oflags...    optimize opcodes, immediates and branch offsets
   -O0        no optimization
   -O1        minimal optimization
   -Ox        multipass optimization (default)
   -Ov        display the number of passes executed at the end
-t            assemble in limited SciTech TASM compatible mode

-E (or -e)    preprocess only (writes output to stdout by default)
-a            don't preprocess (assemble only)
-Ipath        add a pathname to the include file path
-Pfile        pre-include a file (also --include)
```

```
 -Dmacro[=str] pre-define a macro
 -Umacro       undefine a macro
--pragma str   pre-executes a specific %%pragma
--before str   add line (usually a preprocessor statement) before the input
--no-line      ignore %line directives in input

--prefix str   prepend the given string to the names of all extern,
               common and global symbols (also --gprefix)
--suffix str   append the given string to the names of all extern,
               common and global symbols (also --gprefix)
--lprefix str  prepend the given string to local symbols
--lpostfix str append the given string to local symbols

--reproducible attempt to produce run-to-run identical output

 -w+x          enable warning x (also -Wx)
 -w-x          disable warning x (also -Wno-x)
 -w[+-]error   promote all warnings to errors (also -Werror)
 -w[+-]error=x promote warning x to errors (also -Werror=x)
    all                  all possible warnings
    bnd                  invalid BND prefixes [on]
    db-empty             no operand for data declaration [on]
    environment          nonexistent environment variable [on]
    float                all warnings prefixed with "float-"
    float-denorm         floating point denormal [off]
    float-overflow       floating point overflow [on]
    float-toolong        too many digits in floating-point number [on]
    float-underflow      floating point underflow [off]
    hle                  invalid HLE prefixes [on]
    label                all warnings prefixed with "label-"
```

```
label-orphan          labels alone on lines without trailing `:' [on]
label-redef           label redefined to an identical value [off]
label-redef-late      label (re)defined during code generation [error]
lock                  LOCK prefix on unlockable instructions [on]
macro                 all warnings prefixed with "macro-"
macro-def             all warnings prefixed with "macro-def-"
macro-def-case-single single-line macro defined both case sensitive and insensitive [on]
macro-def-greedy-single single-line macro [on]
macro-def-param-single single-line macro defined with and without parameters [error]
macro-defaults        macros with more default than optional parameters [on]
macro-params          all warnings prefixed with "macro-params-"
macro-params-legacy   improperly calling multi-line macro for legacy support [on]
macro-params-multi    multi-line macro calls with wrong parameter count [on]
macro-params-single   single-line macro calls with wrong parameter count [on]
negative-rep          regative %rep count [on]
number-overflow       numeric constant does not fit [on]
obsolete              all warnings prefixed with "obsolete-"
obsolete-nop          instruction obsolete and is a noop on the target CPU [on]
obsolete-removed      instruction obsolete and removed on the target CPU [on]
obsolete-valid        instruction obsolete but valid on the target CPU [on]
phase                 phase error during stabilization [off]
pragma                all warnings prefixed with "pragma-"
pragma-bad            malformed %pragma [off]
pragma-empty          empty %pragma directive [off]
pragma-na             %pragma not applicable to this compilation [off]
pragma-unknown        unknown %pragma facility or directive [off]
ptr                   non-NASM keyword used in other assemblers [on]
regsize               register size specification ignored [on]
unknown-warning       unknown warning in -W/-w or warning directive [off]
user                  %warning directives [on]
```

```
    warn-stack-empty     warning stack empty [on]
    zeroing              RESx in initialized section becomes zero [on]
    zext-reloc           relocation zero-extended to match output format [on]
    other                any warning not specifially mentioned above [on]

--limit-X val  set execution limit X
    passes               total number of passes [unlimited]
    stalled-passes       number of passes without forward progress [1000]
    macro-levels         levels of macro expansion [10000]
    macro-tokens         tokens processed during single-lime macro expansion [10000000]
    mmacros              multi-line macros before final return [100000]
    rep                  %rep count [1000000]
    eval                 expression evaluation descent [8192]
    lines                total source lines processed [2000000000]
```

Отладчик EDB

Этот отладчик есть в составе многих дистрибутивов, например, в Fedora Linux. Тем не менее, имеет смысл его собрать самому.

1) Устанавливаем из дистрибутива необходимые зависимости (**`-devel`**), если они не были установлены ранее:

`Qt5, Boost, Capstone, Graphviz`

В Fedora Linux

```
$ sudo dnf install -y qt5-devel boost-devel capstone-devel graphviz-devel
```

FC33++
qt5-qtbase-devel
qt5-qtxmlpatterns-devel
Qt5SvgConfig.cmake

2) Скачиваем, собираем и устанавливаем библиотеку **gdtoa**

```
$ git clone https://github.com/10110111/gdtoa-desktop.git
$ cd  gdtoa-desktop
$ mkdir build
$ cd  build
$ cmake  ..                                        # cmake -DCMAKE_INSTALL_PREFIX=/usr ..
-- Check for working C compiler: /usr/lib64/ccache/cc
-- Check for working C compiler: /usr/lib64/ccache/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
-- Configuring done
-- Generating done
-- Build files have been written to: /home/podenok/software/gdtoa-desktop/gdtoa-desktop/build
$ make -j8
Scanning dependencies of target arithchk
   ...
Scanning dependencies of target gd_qnan
   ...
Scanning dependencies of target gdtoa-desktop
   ...
Scanning dependencies of target gdtoa_h
[100%] Generating gdtoa-functions-renamed.h
[100%] Built target gdtoa_h
$
```

```
$ sudo make install
[  3%] Built target arithchk
[  7%] Built target gd_qnan
[ 98%] Built target gdtoa-desktop
[100%] Built target gdtoa_h
Install the project...
-- Install configuration: ""
-- Installing: /usr/local/lib64/libgdtoa-desktop.so
-- Installing: /usr/local/include/gdtoa-desktop/arith.h
-- Installing: /usr/local/lib64/pkgconfig/gdtoa-desktop.pc
-- Installing: /usr/local/include/gdtoa-desktop/gdtoa-desktop.h
$
```

3) Скачиваем, собираем и устанавливаем **edb**

```
$ git clone --recurse-submodules https://github.com/eteran/edb-debugger.git
$ cd edb-debugger
$ mkdir build
$ cd  build
$ cmake ..                                      # cmake -DCMAKE_INSTALL_PREFIX=/usr ..
-- The CXX compiler identification is GNU 10.2.1
-- Check for working CXX compiler: /usr/lib64/ccache/c++
-- Check for working CXX compiler: /usr/lib64/ccache/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Found Git: /usr/bin/git (found version "2.26.2")
-- Found PkgConfig: /usr/bin/pkg-config (found version "1.6.3")
-- Checking for module 'capstone>=3.0.4'
--   Found capstone, version 4.0.2
...
-- Configuring done
-- Generating done
-- Build files have been written to: /home/podenok/software/edb/edb-debugger/build
$ make -j8
$ sudo make install
$ edb --version
edb version: 1.3.0
```

```
$ edb --version
edb: error while loading shared libraries: libgdtoa-desktop.so: cannot open shared
object file: No such file or directory
```

Есть два метода решать такие проблемы:

1) LD_LIBRARY_PATH

```
$ LD_LIBRARY_PATH=/usr/local/lib64 edb --version
```

либо экспорт этой переменной в окружение

```
$ export LD_LIBRARY_PATH=/usr/local/lib64
$ edb --version
```

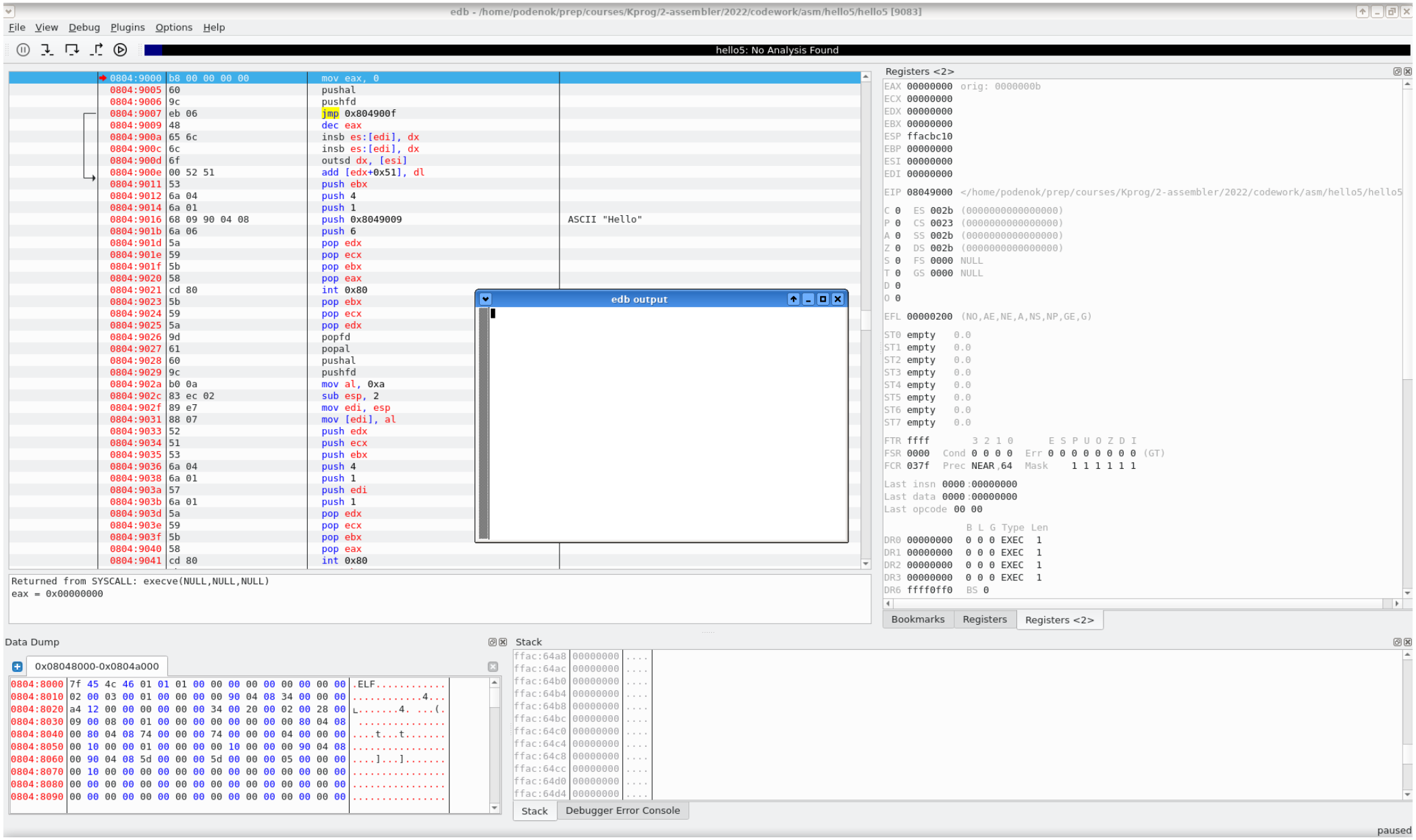
2) ldconfig

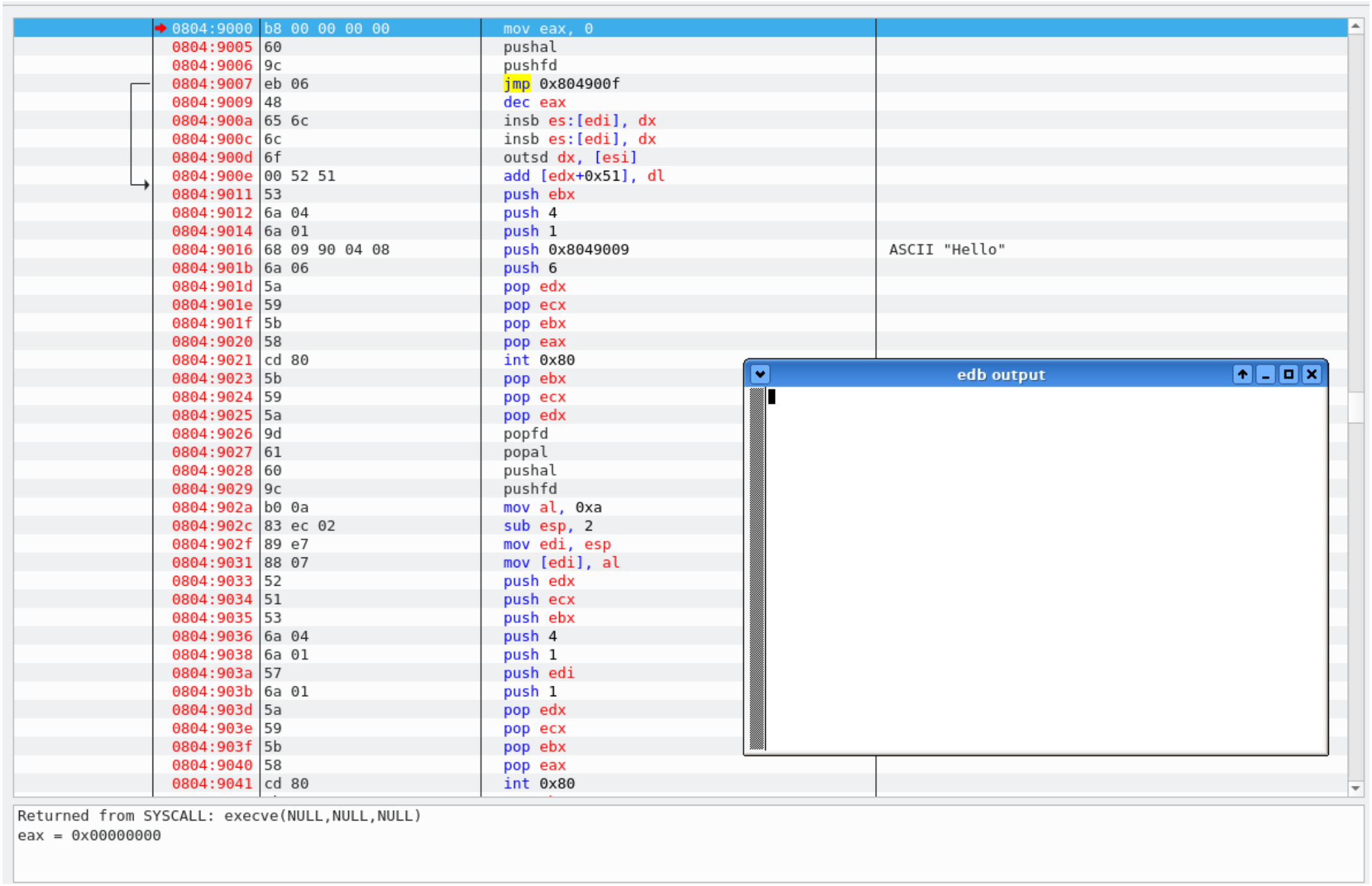
Добавляем в конец файла /etc/ld.so.conf строки

```
/usr/local/lib64
/usr/local/lib
```

или создаем в каталоге /etc/ld.co.conf.d файл с указанными строками

```
$ edb --version
edb version: 1.3.0
```


edb - /home/podenok/prep/courses/Kprog/2-assembler/2022/codework/asm/hello5/hello5 [9083]
File View Debug Plugins Options Help
hello5: No Analysis Found
0804:9000 b8 00 00 00 00 mov eax, 0
0804:9005 60 pushal
0804:9006 9c pushfd
0804:9007 eb 06 jmp 0x804900f
0804:9009 48 dec eax
0804:900a 65 6c insb es:[edi], dx
0804:900c 6c insb es:[edi], dx
0804:900d 6f outsd dx, [esi]
0804:900e 00 52 51 add [edx+0x51], dl
0804:9011 53 push ebx
0804:9012 6a 04 push 4
0804:9014 6a 01 push 1
0804:9016 68 09 90 04 08 push 0x8049009 ASCII "Hello"
0804:901b 6a 06 push 6
0804:901d 5a pop edx
0804:901e 59 pop ecx
0804:901f 5b pop ebx
0804:9020 58 pop eax
0804:9021 cd 80 int 0x80
0804:9023 5b pop ebx
0804:9024 59 pop ecx
0804:9025 5a pop edx
0804:9026 9d popfd
0804:9027 61 popal
0804:9028 60 pushal
0804:9029 9c pushfd
0804:902a b0 0a mov al, 0xa
0804:902c 83 ec 02 sub esp, 2
0804:902f 89 e7 mov edi, esp
0804:9031 88 07 mov [edi], al
0804:9033 52 push edx
0804:9034 51 push ecx
0804:9035 53 push ebx
0804:9036 6a 04 push 4
0804:9038 6a 01 push 1
0804:903a 57 push edi
0804:903b 6a 01 push 1
0804:903d 5a pop edx
0804:903e 59 pop ecx
0804:903f 5b pop ebx
0804:9040 58 pop eax
0804:9041 cd 80 int 0x80
edb output
Returned from SYSCALL: execve(NULL,NULL,NULL)
eax = 0x00000000
Registers <2>
EAX 00000000 orig: 0000000b
ECX 00000000
EDX 00000000
EBX 00000000
ESP ffacbc10
EBP 00000000
ESI 00000000
EDI 00000000
EIP 08049000 </home/podenok/prep/courses/Kprog/2-assembler/2022/codework/asm/hello5/hello5
C 0 ES 002b (0000000000000000)
P 0 CS 0023 (0000000000000000)
A 0 SS 002b (0000000000000000)
Z 0 DS 002b (0000000000000000)
S 0 FS 0000 NULL
T 0 GS 0000 NULL
D 0
O 0
EFL 00000200 (NO,AE,NE,A,NS,NP,GE,G)
ST0 empty 0.0
ST1 empty 0.0
ST2 empty 0.0
ST3 empty 0.0
ST4 empty 0.0
ST5 empty 0.0
ST6 empty 0.0
ST7 empty 0.0
FTR ffff 3 2 1 0 E S P U O Z D I
FSR 0000 Cond 0 0 0 0 Err 0 0 0 0 0 0 0 0 (GT)
FCR 037f Prec NEAR,64 Mask 1 1 1 1 1 1
Last insn 0000:00000000
Last data 0000:00000000
Last opcode 00 00
B L G Type Len
DR0 00000000 0 0 0 EXEC 1
DR1 00000000 0 0 0 EXEC 1
DR2 00000000 0 0 0 EXEC 1
DR3 00000000 0 0 0 EXEC 1
DR6 ffff0ff0 BS 0
Bookmarks Registers Registers <2>
Data Dump
0x08048000-0x0804a000
0804:8000 7f 45 4c 46 01 01 01 00 00 00 00 00 00 00 00 00 .ELF............
0804:8010 02 00 03 00 01 00 00 00 90 04 08 34 00 00 00 00 ....4....
0804:8020 a4 12 00 00 00 00 00 00 34 00 20 00 02 00 28 00 ....4. ...(.
0804:8030 09 00 08 00 01 00 00 00 00 00 00 00 80 04 08 00 ....
0804:8040 00 80 04 08 74 00 00 00 74 00 00 00 04 00 00 00 ....t...t.......
0804:8050 00 10 00 00 01 00 00 00 00 10 00 00 00 90 04 08 ....
0804:8060 00 90 04 08 5d 00 00 00 5d 00 00 00 05 00 00 00 ....]...].......
0804:8070 00 10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
0804:8080 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
0804:8090 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
Stack
ffac:64a8 00000000 ....
ffac:64ac 00000000 ....
ffac:64b0 00000000 ....
ffac:64b4 00000000 ....
ffac:64b8 00000000 ....
ffac:64bc 00000000 ....
ffac:64c0 00000000 ....
ffac:64c4 00000000 ....
ffac:64c8 00000000 ....
ffac:64cc 00000000 ....
ffac:64d0 00000000 ....
ffac:64d4 00000000 ....
Stack Debugger Error Console
paused


0804:9000 b8 00 00 00 00 mov eax, 0
0804:9005 60 pushal
0804:9006 9c pushfd
0804:9007 eb 06 jmp 0x804900f
0804:9009 48 dec eax
0804:900a 65 6c insb es:[edi], dx
0804:900c 6c insb es:[edi], dx
0804:900d 6f outsd dx, [esi]
0804:900e 00 52 51 add [edx+0x51], dl
0804:9011 53 push ebx
0804:9012 6a 04 push 4
0804:9014 6a 01 push 1
0804:9016 68 09 90 04 08 push 0x8049009
ASCII "Hello"
0804:901b 6a 06 push 6
0804:901d 5a pop edx
0804:901e 59 pop ecx
0804:901f 5b pop ebx
0804:9020 58 pop eax
0804:9021 cd 80 int 0x80
0804:9023 5b pop ebx
0804:9024 59 pop ecx
0804:9025 5a pop edx
0804:9026 9d popfd
0804:9027 61 popal
0804:9028 60 pushal
0804:9029 9c pushfd
0804:902a b0 0a mov al, 0xa
0804:902c 83 ec 02 sub esp, 2
0804:902f 89 e7 mov edi, esp
0804:9031 88 07 mov [edi], al
0804:9033 52 push edx
0804:9034 51 push ecx
0804:9035 53 push ebx
0804:9036 6a 04 push 4
0804:9038 6a 01 push 1
0804:903a 57 push edi
0804:903b 6a 01 push 1
0804:903d 5a pop edx
0804:903e 59 pop ecx
0804:903f 5b pop ebx
0804:9040 58 pop eax
0804:9041 cd 80 int 0x80
edb output
Returned from SYSCALL: execve(NULL,NULL,NULL)
eax = 0x00000000


Registers <2>
EAX 00000000 orig: 0000000b
ECX 00000000
EDX 00000000
EBX 00000000
ESP ffacbc10
EBP 00000000
ESI 00000000
EDI 00000000
EIP 08049000 </home/podenok/prep/courses/Kprog/2-assembler/2022/codework/asm/hello5/hello5
C 0 ES 002b (0000000000000000)
P 0 CS 0023 (0000000000000000)
A 0 SS 002b (0000000000000000)
Z 0 DS 002b (0000000000000000)
S 0 FS 0000 NULL
T 0 GS 0000 NULL
D 0
O 0
EFL 00000200 (NO,AE,NE,A,NS,NP,GE,G)
ST0 empty 0.0
ST1 empty 0.0
ST2 empty 0.0
ST3 empty 0.0
ST4 empty 0.0
ST5 empty 0.0
ST6 empty 0.0
ST7 empty 0.0
FTR ffff 3 2 1 0 E S P U O Z D I
FSR 0000 Cond 0 0 0 0 Err 0 0 0 0 0 0 0 0 (GT)
FCR 037f Prec NEAR,64 Mask 1 1 1 1 1 1
Last insn 0000:00000000
Last data 0000:00000000
Last opcode 00 00
B L G Type Len
DR0 00000000 0 0 0 EXEC 1
DR1 00000000 0 0 0 EXEC 1
DR2 00000000 0 0 0 EXEC 1
DR3 00000000 0 0 0 EXEC 1
DR6 ffff0ff0 BS 0
Bookmarks
Registers
Registers <2>

Data Dump

0x08048000-0x0804a000

0804:8000	7f 45 4c 46 01 01 01 00 00 00 00 00 00 00 00 00	.ELF....	
0804:8010	02 00 03 00 01 00 00 00 90 04 08 34 00 00 00	....	4...
0804:8020	a4 12 00 00 00 00 00 00 34 00 20 00 02 00 28 00	....	4. ...(.
0804:8030	09 00 08 00 01 00 00 00 00 00 00 00 80 04 08	....	
0804:8040	00 80 04 08 74 00 00 00 74 00 00 00 04 00 00 00	....t...	t...
0804:8050	00 10 00 00 01 00 00 00 00 10 00 00 00 90 04 08	....	
0804:8060	00 90 04 08 5d 00 00 00 5d 00 00 00 05 00 00 00	....]...	]...
0804:8070	00 10 00 00 00 00 00 00 00 00 00 00 00 00 00 00	....	
0804:8080	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	....	
0804:8090	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	....	

Stack

ffac:bc10	00000001	....	
ffac:bc14	ffacd265	e...	ASCII "hello5"
ffac:bc18	00000000	....	
ffac:bc1c	ffacd26c	l...	ASCII "SHELL=/bin/bash"
ffac:bc20	ffacd27c	|...	ASCII "SESSION_MANAGER=local/unix:@/tmp/.ICE-unix/1356,unix/unix:/tmp/.ICE-unix/1356"
ffac:bc24	ffacd2ca	...	ASCII "WINDOWID=27262979"
ffac:bc28	ffacd2dc	|...	ASCII "COLORTERM=truecolor"
ffac:bc2c	ffacd2f0	...	ASCII "XDG_CONFIG_DIRS=/etc/xdg"
ffac:bc30	ffacd309	...	ASCII "XDG_SESSION_PATH=/org/freedesktop/DisplayManager/Session1"
ffac:bc34	ffacd343	C...	ASCII "HISTCONTROL=ignoreboth"
ffac:bc38	ffacd35a	Z...	ASCII "XDG_MENU_PREFIX=xfce-"
ffac:bc3c	ffacd370	p...	ASCII "HOSTNAME=logovo"

Stack | Debugger Error Console

paused

Рабочая среда практических занятий

- ассемблер **nasm** (nasm);
- компилятор языка **C** (gcc);
- утилита **make** (make);
- отладчик **gdb** (gdb);
- отладчик **edb** (edb);
- компоновщик **ld** (binutils);
- компоновщик **gcc** (gcc);
- редактор текста или IDE, поддерживающие подсветку синтаксиса (slickedit).

Файлы **stud_io.inc** из архива материалов к книге «Столяров А.В. Программирование: Введение в профессию. Т.2. Низкоуровневое программирование. 2016.pdf» (на ftp)

Содержимое каталога простой программы, с которой имеет смысл начать

```
$ ls -1
makefile
stud_io.inc
hello5.asm
```

Пример программы

```
$ cat hello5.asm
%include "stud_io.inc"     ; директива препроцессора
global   _start            ; директива ассемблера

section  .text             ; директива ассемблера
_start:                    ; метка
          mov      eax, 0
again:
          PRINT    "Hello" ; макрос (макрокоманда)
          PUTCHAR  10      ; макрос
          inc      eax
          cmp      eax, 5
          jl       again
          FINISH           ; макрос
```

Содержимое файла makefile (конкретно под программу hello5)

```
#  makefile
all: hello5
# правила сборки
hello5: hello5.o
	ld -m elf_i386 -o hello5 hello5.o

# правила ассемблирования
hello5.o: hello5.asm stud_io.inc makefile
	nasm -Wall -f elf -gdwarf -l hello5.lst -o hello5.o hello5.asm

.PHONY: clean
clean:
	rm -f hello5.o hello5.lst
```

Содержимое файла makefile (более общее)

```
$ cat makefile
NAME=hello5
INCLUDES=stud_io.inc

AS=nasm
CC=gcc
#LD=gcc                                  # LD=ld по-умолчанию

LDFLAGS=-m elf_i386                      # -m32 для LD=gcc
ASFLAGS=-Wall -f elf -g

.SUFFIXES:
.SUFFIXES: .o .c .asm

all: $(NAME)

$(NAME): $(NAME).o
        $(LD) $(LDFLAGS) $^ -o $@

.PHONY: clean
clean:
        $(RM) $(NAME) *.o *.lst

$(NAME).o: $(NAME).asm $(INCLUDES) makefile
        $(AS) $(ASFLAGS) -l $(*F).lst $< -o $@
```

Ассемблерная строка NASM

Програма на ассемблере состоит из набора строк в алфавите подмножества печатных символов ASCII.

Каждая ассемблерная строка представляет собой одно из:

1) директива препроцессора;
2) директива ассемблера;
3) инструкция;
4) макрос;
5) комментарий.

Строка инструкции имеет следующий формат:

```
метка[:] [<LF>] инструкция [операнды] [; текст_комментария]
```

<*LF*> — перенос строки, т.е. инструкция может быть записана и так:

```
метка[:]
        инструкция [операнды] [; текст_комментария]
; комментарий_также_может_быть_на_отдельной_строке
```

Обязательным полем является только поле инструкции.

Необходимость поля операндов определяется инструкцией процессора.

NASM использует символ `'\'` в качестве символа продолжения строки — если строка заканчивается символом `'\'`, следующая строка считается частью строки с символом `'\'` на конце.

Пример простой программы

```
$ cat foo.asm
section         .data                   ; (d – директива ассемблера) секция данных RW
a               dw      185             ; (d)

section         .text                   ; (d) секция кода
global          _start                  ; (d)

_start:                                 ; (L -- метка)
                xor     eax, eax        ; (i -- инструкция)
                mov     ax, [a]         ; (i)
                not     ax              ; (i)
                add     ax, 1           ; (i)
fin:                                    ; (L)
                mov     eax, 1          ; (i)
                mov     ebx, 0          ; (i)
                int     0x80            ; (i)
```

NASM не накладывает ограничений на количество пробелов в строке – метки могут иметь пробелы вначале, а перед инструкцией пробелы не обязательны:

```
      _start:
xor    eax, eax
mov    ax, [a]
```

Листинг простой программы

```
$ cat foo.list
     1                                  section         .data
     2 00000000 B900                    a               dw      185
     3
     4                                  section         .text
     5                                  global          _start
     6
     7                                  _start:
     8                                  main:
     9 00000000 31C0                                    xor     eax, eax
    10 00000002 66A1[00000000]                          mov     ax, [a]
    11 00000008 66F7D0                                  not     ax
    12 0000000B 6683C001                                add     ax, 1
    13                                  fin:
    14 0000000F B801000000                              mov     eax, 1
    15 00000014 BB00000000                              mov     ebx, 0
    16 00000019 CD80                                    int     0x80
```

Практика программирования

```
    13 0000000F B801000000                              mov     eax, 1
    14 00000014 BB00000000                              mov     ebx, 0
    15 00000019 CD80                                    int     0x80
или
    13 0000000F 31C0                                    xor     eax, eax
    14 00000011 40                                      inc     eax
    15 00000012 31DB                                    xor     ebx, ebx
    16 00000014 CD80                                    int     0x80
```

Пример программы с макросами

```
$ cat hello5.asm
%include "stud_io.inc"     ; директива препроцессора
global   _start            ; директива ассемблера

section  .text             ; директива ассемблера
_start:                    ; метка "точка входа"
          mov      eax, 0
again:
          PRINT    "Hello" ; макрос (макрокоманда)
          PUTCHAR  10      ; макрос
          inc      eax
          cmp      eax, 5
          jl       again
          FINISH           ; макрос
```

Практика программирования

```
_start:                    ; метка "точка входа"
          mov      ecx, 5  ; счетчик
again:
          PRINT    "Hello" ; макрос (макрокоманда)
          PUTCHAR  10      ; макрос
          dec      ecx
          jnz      again
          FINISH           ; макрос
```

```
$ nasm -f elf -l hello5.lst hello5.asm
$ less hello5.lst
 1                      %include        "stud_io.inc"
 1                  <1> ;; File stud_io.inc for both Linux and FreeBSD.
 2                  <1> ;; Copyright (c) Andrey Vikt. Stolyarov, 2009, 2015
 3                  <1> ;; I, the author, hereby grant everyone the right to use this
 4                  <1> ;; file for any purpose, in any manner, in it's original or
 5                  <1> ;; modified form, provided that any modified versions are
 6                  <1> ;; clearly marked as such.
... < Пропущена часть, включаемая директивой %include >
 2                              global  _start                  ; директива ассемблера
 3
 4                              section .text                   ; директива ассемблера
 5                              _start:                         ; метка
 6 00000000 B800000000                          mov     eax, 0
 6 00000000 31C0                                xor     eax, eax ; add mod 2
 7                              again:
 8                                              PRINT   "Hello"  ; макрос (макрокоманда)
 8 00000005 60                  <1>  pusha
 8 00000006 9C                  <1>  pushf
 8 00000007 EB06                <1>  jmp %%astr
 8 00000009 48656C6C6F00        <1> %%str db %1, 0
           H e l l o \0
 8                              <1> %%strln equ $-%%str
 8                              <1> %%astr: _syscall_write 1, %%str, %%strln
 8                              <2> ..@2.astr:
 8                              <2>  _syscall_3 4,%1,%2,%3
 8 0000000F 52                  <3>  push edx
 8 00000010 51                  <3>  push ecx
 8 00000011 53                  <3>  push ebx
```

```
 8 00000012 6A04           <3>  push %1
 8 00000014 6A01           <3>  push %2
 8 00000016 68[09000000]   <3>  push %3
 8 0000001B 6A06           <3>  push %4
 8 0000001D 5A             <3>  pop edx
 8 0000001E 59             <3>  pop ecx
 8 0000001F 5B             <3>  pop ebx
 8 00000020 58             <3>  pop eax
 8 00000021 CD80           <3>  int 0x80
 8 00000023 5B             <3>  pop ebx
 8 00000024 59             <3>  pop ecx
 8 00000025 5A             <3>  pop edx
 8 00000026 9D             <1>  popf
 8 00000027 61             <1>  popa
 9                                     PUTCHAR  10       ; макрос
 9 00000028 60             <1>  pusha
 9 00000029 9C             <1>  pushf
 9                         <1> %ifstr %1
 9                         <1>  mov al, %1
 9                         <1> %elifnum %1
 9 0000002A B00A           <1>  mov al, %1
 9                         <1> %elifidni %1,al
 9                         <1>  nop
 9                         <1> %elifidni %1,ah
 9                         <1>  mov al, ah
 9                         <1> %elifidni %1,bl
 9                         <1>  mov al, bl
 9                         <1> %elifidni %1,bh
 9                         <1>  mov al, bh
 9                         <1> %elifidni %1,cl
```

```
9                                       <1>  mov al, cl
9                                       <1> %elifidni %1,ch
9                                       <1>  mov al, ch
9                                       <1> %elifidni %1,dl
9                                       <1>  mov al, dl
9                                       <1> %elifidni %1,dh
9                                       <1>  mov al, dh
9                                       <1> %else
9                                       <1>  mov al, %1
9                                       <1> %endif
9 0000002C 83EC02                       <1>  sub esp, 2
9 0000002F 89E7                         <1>  mov edi, esp
9 00000031 8807                         <1>  mov [edi], al
9                                       <1>  _syscall_write 1, edi, 1
9                                       <2>  _syscall_3 4,%1,%2,%3
9 00000033 52                           <3>  push edx
9 00000034 51                           <3>  push ecx
9 00000035 53                           <3>  push ebx
9 00000036 6A04                         <3>  push %1
9 00000038 6A01                         <3>  push %2
9 0000003A 57                           <3>  push %3
9 0000003B 6A01                         <3>  push %4
9 0000003D 5A                           <3>  pop edx
9 0000003E 59                           <3>  pop ecx
9 0000003F 5B                           <3>  pop ebx
9 00000040 58                           <3>  pop eax
9 00000041 CD80                         <3>  int 0x80
9 00000043 5B                           <3>  pop ebx
9 00000044 59                           <3>  pop ecx
9 00000045 5A                           <3>  pop edx
```

```
 9 00000046 83C402          <1>  add esp, 2
 9 00000049 9D              <1>  popf
 9 0000004A 61              <1>  popa
10 0000004B 40                        inc     eax     ; инкремент
11 0000004C 83F805                    cmp     eax, 5  ; compare
12 0000004F 7CB4                      jl      again   ; jump if less
13                                    FINISH          ; макрос
13                         <1>  _syscall_exit %1
13 00000051 BB00000000     <2>  mov ebx, %1
13 00000056 B801000000     <2>  mov eax, 1
13 0000005B CD80           <2>  int 0x80
$
```

Формат строки инструкции

метка[:] [<LF>] инструкция [операнды] [; текст_комментария]

Метки

Двоеточие после метки необязательно.

Это означает, что если в строке вместо инструкции **lodsb** будет ошибочно указано **lodab**, строка останется корректной, но вместо инструкции будет объявлена метка.

Выявить такие опечатки можно, используя опцию **-w+orphan-labels**, которая в случае обнаружении метки без заключительного двоеточия выдаст предупреждение.

```
foo.asm:8: warning: label alone on a line without a colon might be in error [-w+orphan-labels]
```

Допустимыми символами в метках являются:

- буквы [**A-Za-z**]
- цифры [**0-9**]
- знаки [**_~@#$.?**]

Допустимые символы в начале метки (первый символ метки) — только:

- буквы [**A-Za-z**]
- знаки [**_.?$**]

В идентификаторе может также присутствовать префикс **$** для указания того, что это действительно идентификатор, а не зарезервированное слово — таким образом, если в некотором модуле описан символ **eax**, в коде NASM для указания того, что это не регистр, следует на него сослаться как **$eax**.

Инструкции

Поле инструкций может содержать любые процессорные инструкции — поддерживаются инструкции Pentium и P6, FPU, MMX, ... а также некоторые недокументированные инструкции.

- перед инструкциями могут присутствовать префиксы

```
LOCK,
REP, REPE/REPZ, REPNE/REPNZ
XACQUIRE/XRELEASE
BND/NOBND
```

- поддерживаются префиксы размера адреса и операнда

```
A16, A32, A64
O16, O32, O64
```

- в качестве префикса инструкции можно использовать обозначение сегментного регистра

```
es mov [bx], ax
```

что эквивалентно коду

```
mov [es:bx], ax
```

В общем случае рекомендуется использовать последнюю форму, поскольку она согласуется с другими синтаксическими особенностями языка.

Однако, для инструкций, не имеющих операндов (например, **LODS, STOS, ...**) и требующих в некоторых случаях замены сегмента, на данный момент не существует никакого синтаксического способа обойти конструкцию **ES LODS**.

Префиксы переопределения сегмента **CS**, **DS**, **ES**, **SS**, **FS**, **GS**, могут присутствовать в строке самостоятельно. При этом NASM будет генерировать соответствующие префикс-байты.

Префиксы размера адреса и операнда **A16**, **A32**, **A64**, **O16**, **O32**, **O64** также могут быть указаны, но с учетом выходного формата. Например, для формата `elf32` префиксы **A64** и **O64** недопустимы, а **A32** и **O32** не генерируют префикс-байтов.

LOCK или **REP**, также могут присутствовать в строке самостоятельно. При этом NASM будет генерировать соответствующие префикс-байты.

Если для инструкции префикс не предполагается, будет выведено предупреждение

```
foo.asm:8: warning: instruction is not lockable [-w+lock]
foo.asm:12: warning: xacquire invalid with this instruction [-w+hle]
foo.asm:13: warning: xrelease invalid with this instruction [-w+hle]
```

Некоторые префиксы генерируют ошибку

```
foo.asm:14: error: bnd prefix is not allowed
```

Операнды

Операнды инструкций могут принимать несколько форм:

- они могут быть регистрами, например **EAX**, **EBP**, **EBX**, **CR0**. Для обозначения регистров NASM использует синтаксис INTEL;
- эффективными адресами (EA);
- константами;
- выражениями.