# Интегрированная среда разработки и отладки программ MC Studio.

## Инструменты RISCore32

## Руководство программиста

Листов 54

# Порядок использования данного документа

Настоящая документация охраняется действующим законодательством Российской Федерации об авторском праве и смежных правах, в частности, законом Российской Федерации «Об авторском праве и смежных правах». ГУП НПЦ «ЭЛВИС» является единственным правообладателем исключительных авторских прав на настоящую документацию.

Настоящую документацию, не иначе как по предварительному согласию ГУП НПЦ «ЭЛВИС», запрещается:

- воспроизводить, т.е. изготавливать один или более экземпляров настоящей документации, ее части, в любой форме, любым способом;
- сдавать в прокат;
- публично показывать, исполнять или сообщать для всеобщего сведения,
- переводить;
- переделывать или другим образом перерабатывать (дорабатывать).

ГУП НПЦ «ЭЛВИС» оставляет за собой право в любой момент вносить изменения (дополнения) в настоящую документацию без предварительного уведомления о таком изменении (дополнении).

ГУП НПЦ «ЭЛВИС» не несет ответственности за вред, причиненный при использовании настоящей документации.

Передача настоящей документации не означает передачи каких-либо авторских прав ГУП НПЦ «ЭЛВИС» на нее.

Возникновение каких-либо прав на материальный носитель, на котором передается настоящая документация, не влечет передачи каких-либо авторских прав на данную документацию.

Все указанные в настоящей документации товарные знаки принадлежат их владельцам.

# ОГЛАВЛЕНИЕ

1. ВВЕДЕНИЕ ..... 8
2. КОМПИЛЯТОР С ..... 9
2.1. ВВЕДЕНИЕ В С-КОМПИЛЯТОР ..... 9
2.2. КЛЮЧИ С-КОМПИЛЯТОРА ..... 9
2.2.1. -c ..... 10
2.2.2. -S ..... 10
2.2.3. -o ..... 10
2.2.4. -D ..... 11
2.2.5. -U ..... 11
2.2.6. -E ..... 11
2.2.7. -I ..... 11
2.2.8. -L ..... 11
2.2.9. -I- ..... 11
2.2.10. -Wa ..... 11
2.2.11. -Wl ..... 11
2.2.12. -include ..... 11
2.2.13. -imacros ..... 11
2.2.14. -idirafter ..... 12
2.2.15. -iprefix ..... 12
2.2.16. -iwithprefix ..... 12
2.2.17. -iwithprefixbefore ..... 12
2.2.18. -nostdinc ..... 12
2.2.19. -g ..... 12
2.2.20. -O ..... 12
2.2.21. -save-temps ..... 12
2.2.22. -M ..... 12
2.2.23. -MM ..... 12
2.2.24. -MD ..... 13
2.2.25. -MMD ..... 13
2.2.26. -MG ..... 13
2.2.27. -H ..... 13
2.2.28. -pipe ..... 13
2.2.29. -time ..... 13
2.2.30. -B ..... 13
2.2.31. -V ..... 13
2.2.32. -mcpu ..... 13
2.2.33. -mipsX ..... 14
2.2.34. -mfp32 ..... 14
2.2.35. -mfp64 ..... 14
2.2.36. -mgp32 ..... 14
2.2.37. -mgp64 ..... 14
2.2.38. -mint64 ..... 14
2.2.39. -mlong64 ..... 14
2.2.40. -mlong32 ..... 14
2.2.41. -mabiX ..... 15
2.2.42. -mmips-as ..... 15
2.2.43. -mgas ..... 15
2.2.44. -msplit-addresses ..... 15
2.2.45. -mno-split-addresses ..... 15
2.2.46. -mrnames ..... 15
2.2.47. -mno-rnames ..... 15
2.2.48. -mgpopt ..... 15
2.2.49. -mno-gpopt ..... 15
2.2.50. -mstats ..... 15
2.2.51. -mno-stats ..... 16
2.2.52. -mmemcpy ..... 16
2.2.53. -mno-memcpy ..... 16
2.2.54. -mmips-tfile ..... 16
2.2.55. -mno-mips-tfile ..... 16
2.2.56. -msoft-float ..... 16
2.2.57. -mhard-float ..... 16
2.2.58. -mabicalls ..... 16
2.2.59. -mnoabi-calls ..... 16
2.2.60. -mlong-calls ..... 16

2.2.61. -mno-long-calls ..... 16
2.2.62. -mhalf-pic ..... 17
2.2.63. -mno-half-pic ..... 17
2.2.64. -membedded-pic ..... 17
2.2.65. -mno-embedded-pic ..... 17
2.2.66. -membedded-data ..... 17
2.2.67. -mno-embedded-data ..... 17
2.2.68. -msingle-float ..... 17
2.2.69. -mdouble-float ..... 17
2.2.70. -m4650 ..... 17
2.2.71. -mips16 ..... 17
2.2.72. -mno-mips16 ..... 17
2.2.73. -mentry ..... 18
2.2.74. -EL ..... 18
2.2.75. -EB ..... 18
2.2.76. -nocpp ..... 18
3. АССЕМБЛЕР ..... 19
3.1. АССЕМБЛЕР ..... 19
3.2. КЛЮЧИ КОМАНДНОЙ СТРОКИ АССЕМБЛЕРА ..... 19
3.2.1. -a ..... 20
3.2.2. -D ..... 20
3.2.3. --defsym ..... 20
3.2.4. --em ..... 20
3.2.5. -f ..... 20
3.2.6. --gstabs ..... 20
3.2.7. --gdwarf2 ..... 20
3.2.8. --help ..... 20
3.2.9. -I ..... 21
3.2.10. -J ..... 21
3.2.11. -K ..... 21
3.2.12. -L ..... 21
3.2.13. --MD ..... 21
3.2.14. -o ..... 21
3.2.15. -R ..... 21
3.2.16. --statistics ..... 21
3.2.17. --strip-local-absolute ..... 21
3.2.18. --traditional-format ..... 22
3.2.19. --version ..... 22
3.2.20. -W ..... 22
3.2.21. --warn ..... 22
3.2.22. --fatal-warnings ..... 22
3.2.23. --itbl ..... 22
3.2.24. -Z ..... 22
3.2.25. --listing-lhs-width ..... 22
3.2.26. --listing-lhs-width2 ..... 22
3.2.27. --listing-rhs-width ..... 22
3.2.28. --listing-cont-lines ..... 22
3.2.29. -membedded-pic ..... 22
3.2.30. -EB ..... 23
3.2.31. -EL ..... 23
3.2.32. -G ..... 23
3.2.33. --trap ..... 23
3.2.34. --break ..... 23
3.2.35. --KPIC ..... 23
3.2.36. -non_shared ..... 23
3.2.37. -xgot ..... 23
3.2.38. -32 ..... 23
3.2.39. -64 ..... 23
3.2.40. -mNNNN ..... 23
3.2.41. -no-mNNNN ..... 23
3.2.42. --construct-floats ..... 24
3.2.43. --no-construct-floats ..... 24
3.3. РАБОТА АССЕМБЛЕРА ..... 24
3.4. ФОРМАТ ИСХОДНОГО ФАЙЛА АССЕМБЛЕРА ..... 24
3.5. СООБЩЕНИЯ ОБ ОШИБКАХ И ПРЕДУПРЕЖДЕНИЯ ..... 24
3.6. УСЛОВНОЕ АССЕМБЛИРОВАНИЕ ..... 25

3.7. ДИРЕКТИВЫ АССЕМБЛЕРА ..... 25
3.7.1. .abort ..... 27
3.7.2. .align ..... 27
3.7.3. .ascii ..... 27
3.7.4. .asciz ..... 27
3.7.5. .balign ..... 27
3.7.6. .byte ..... 27
3.7.7. .comm ..... 27
3.7.8. .data ..... 28
3.7.9. .def ..... 28
3.7.10. .desc ..... 28
3.7.11. .eject ..... 28
3.7.12. .else ..... 28
3.7.13. .end ..... 28
3.7.14. .endef ..... 28
3.7.15. .endif ..... 28
3.7.16. .endr ..... 28
3.7.17. .equ ..... 28
3.7.18. .ent ..... 29
3.7.19. .extern ..... 29
3.7.20. .file ..... 29
3.7.21. .fill ..... 29
3.7.22. .global ..... 29
3.7.23. .hword ..... 29
3.7.24. .ident ..... 29
3.7.25. .if ..... 30
3.7.26. .ifdef ..... 30
3.7.27. .ifnotdef ..... 30
3.7.28. .include ..... 30
3.7.29. .int ..... 30
3.7.30. .irp ..... 30
3.7.31. .irpc ..... 30
3.7.32. .lcomm ..... 31
3.7.33. .line ..... 31
3.7.34. .ln ..... 31
3.7.35. .list ..... 31
3.7.36. .long ..... 31
3.7.37. .macro ..... 31
3.7.38. .endm ..... 32
3.7.39. .exitm ..... 32
3.7.40. \@ ..... 32
3.7.41. .nolist ..... 32
3.7.42. .octa ..... 32
3.7.43. .p2align ..... 32
3.7.44. .psize ..... 32
3.7.45. .purgem ..... 32
3.7.46. .quad ..... 32
3.7.47. .rept ..... 33
3.7.48. .sbttl ..... 33
3.7.49. .section ..... 33
3.7.50. .set ..... 33
3.7.51. .short ..... 33
3.7.52. .size ..... 33
3.7.53. .space ..... 33
3.7.54. .stab ..... 34
3.7.55. .string ..... 34
3.7.56. .tag ..... 34
3.7.57. .text ..... 35
3.7.58. .title ..... 35
3.7.59. .type ..... 35
3.7.60. .val ..... 35
3.7.61. .word ..... 35
3.8. РЕЗУЛЬТАТ АССЕМБЛИРОВАНИЯ ..... 35
4. ДВОИЧНЫЕ УТИЛИТЫ ..... 36
4.1. ДВОИЧНЫЕ УТИЛИТЫ ..... 36
4.2. КЛЮЧИ ПРОГРАММЫ MDUMP ..... 36
4.3. КЛЮЧИ ПРОГРАММЫ MCOPY ..... 37

4.4. КЛЮЧИ БИБЛИОТЕКАРЯ ..... 38
4.5. NM ..... 39
5. РЕДАКТОР СВЯЗЕЙ LD ..... 41
5.1. ВВЕДЕНИЕ В MLD ..... 41
5.2. КЛЮЧИ КОМАНДНОЙ СТРОКИ КОМПОНОВЩИКА ..... 41
5.2.1. -A ..... 42
5.2.2. -b ..... 42
5.2.3. -c ..... 42
5.2.4. -d ..... 42
5.2.5. -e ..... 42
5.2.6. -E ..... 43
5.2.7. -EB ..... 43
5.2.8. -EL ..... 43
5.2.9. -g ..... 43
5.2.10. -G ..... 43
5.2.11. -l ..... 43
5.2.12. -L ..... 43
5.2.13. -m ..... 43
5.2.14. -M ..... 43
5.2.15. -N ..... 43
5.2.16. -o ..... 44
5.2.17. -O ..... 44
5.2.18. -r ..... 44
5.2.19. -s ..... 44
5.2.20. -S ..... 44
5.2.21. -t ..... 44
5.2.22. -T ..... 44
5.2.23. -u ..... 44
5.2.24. -v ..... 44
5.2.25. -V ..... 44
5.2.26. -x ..... 44
5.2.27. -X ..... 45
5.2.28. --discard-none ..... 45
5.2.29. -y ..... 45
5.2.30. -( ..... 45
5.2.31. -) ..... 45
5.2.32. -BDynamic ..... 45
5.2.33. -BStatic ..... 45
5.2.34. --check-sections ..... 45
5.2.35. --no-check-sections ..... 46
5.2.36. --cref ..... 46
5.2.37. --defsym ..... 46
5.2.38. --demangle ..... 46
5.2.39. --gc-sections ..... 46
5.2.40. --no-gc-sections ..... 46
5.2.41. --help ..... 46
5.2.42. -Map ..... 46
5.2.43. --no-demangle ..... 46
5.2.44. --no-keep-memory ..... 46
5.2.45. --no-undefined ..... 46
5.2.46. --noinhibit-exec ..... 46
5.2.47. --oformat ..... 47
5.2.48. --retain-symbols-file ..... 47
5.2.49. -rpath ..... 47
5.2.50. -rpath-link ..... 47
5.2.51. -shared ..... 47
5.2.52. --sort-common ..... 47
5.2.53. --split-by-file ..... 47
5.2.54. --stats ..... 47
5.2.55. --traditional-format ..... 47
5.2.56. -Ttext ..... 47
5.2.57. -Tdata ..... 47
5.2.58. -Tbss ..... 47
5.2.59. --verbose ..... 47
5.2.60. --version-script ..... 48
5.2.61. --warn-common ..... 48
5.2.62. --warn-multiple-gp ..... 48

5.2.63. --warn-once ..... 48
5.2.64. --warn-section-align ..... 48
5.2.65. --whole-archive ..... 48
5.2.66. --wrap ..... 48
5.3. УПРАВЛЯЮЩИЙ ФАЙЛ КОМПОНОВЩИКА ..... 48
5.4. ДИРЕКТИВЫ КОМПОНОВЩИКА ..... 49
5.4.1. SECTIONS ..... 49
5.4.2. INCLUDE ..... 50
5.4.3. FILE ..... 50
5.4.4. GROUP ..... 50
5.4.5. OUTPUT ..... 50
5.4.6. SEARCH_DIR ..... 50
5.4.7. STARTUP ..... 50
5.4.8. OUTPUT_FORMAT ..... 50
5.4.9. TARGET ..... 50
5.4.10. ASSERT ..... 50
5.4.11. EXTERN ..... 50
5.4.12. NOCROSSREFS ..... 51
5.4.13. OUTPUT_ARCH ..... 51
5.4.14. PROVIDE ..... 51
5.4.15. MEMORY ..... 51
6. ПРЕДМЕТНЫЙ УКАЗАТЕЛЬ ..... 52

# 1. Введение

В данной книге описаны следующие инструменты RISC, используемые для сборки и редактирования файлов проекта:

- компилятор C;
- ассемблер RISC;
- двоичные утилиты;
- компоновщик ld.

# 2. Компилятор С

## 2.1. Введение в С-компилятор

**Компилятор С** для ядра RISC (**mipsel-elf32-gcc)** основан на коде **gcc** и поддерживает все возможности стандарта **ANSI-C**.

Запуск **компилятора С** из командной строки: **mgcc {ключи|файлы}…**
В списке файлов можно указывать *С*-файлы, ассемблерные файлы, объектные файлы, библиотеки. По умолчанию делается попытка скомпилировать и далее собрать все указанные файлы в выполняемый файл. По умолчанию имя файла *a.out*.

## 2.2. Ключи С-компилятора

**С-компилятор** имеет следующие основные ключи:

- -c;
- -S;
- -o;
- -D;
- -U;
- -E;
- -I;
- -L;
- -I-;
- -Wa;
- -Wl;
- -include;
- -imacros;
- -idirafter;
- -iprefix;
- -iwithprefix;
- -iwithprefixbefore;
- -nostdinc;
- -g;
- -O;
- -save-temps;
- -M;
- -MM;
- -MD;
- -MMD;
- -MG;
- -H;
- -pipe;
- -time;
- -B;
- -V.

Специфические ключи:

- -mcpu;
- -mipsX;
- -mfp32;
- -mfp64;
- -mgp32;

- -mgp64;
- -mint64;
- -mlong64;
- -mlong32;
- -mabiX;
- -mmips-as;
- -mgas;
- -msplit-addresses;
- -mno-split-addresses;
- -mrnames;
- -mno-rnames;
- -mgpopt;
- -mno-gpopt;
- -mstats;
- -mno-stats;
- -mmemcpy;
- -mno-memcpy;
- -mmips-tfile;
- -mno-mips-tfile;
- -msoft-float;
- -mhard-float;
- -mabicalls;
- -mno-abicalls;
- -mlong-calls;
- -mno-long-calls;
- -mhalf-pic;
- -mno-half-pic;
- -membedded-pic;
- -mno-embedded-pic;
- -membedded-data;
- -mno-embedded-data;
- -msingle-float;
- -mdouble-float;
- -m4650;
- -mips16;
- -mno-mips16;
- -mentry;
- -EL;
- -EB;
- -nocpp.

### 2.2.1. -c

**Ключ -c** указывает gcc, что результатом компиляции должен быть объектный файл. По умолчанию gcc пытается собрать программу.

### 2.2.2. -S

**Ключ -S** указывает gcc, что результирующим файлом должен быть файл Ассемблера RISC. По умолчанию, имя файла с ассемблерным кодом получается из имени исходного файла заменой суффикса '*.c*', '*.i*', и т.д. на '*.s*'.

### 2.2.3. -o

**Ключ -o file** указывает gcc, что результат работы должен быть записан в файл **file**.

### 2.2.4. -D

**Ключ -Dname[=определение]** задает переменную с именем **name** (эквивалентно использованию *#define*). Это может быть использовано, в частности, для задания условий компиляции (с использованием *#ifdef **name***).

### 2.2.5. -U

**Ключ -Uname** указывает gcc сделать макроопределение **name** неопределенным.

### 2.2.6. -E

**Ключ -E** указывает вывести на стандартный вывод результат препроцессирования (выполняется только препроцессор C) исходных текстов.

### 2.2.7. -I

**Ключ -Idir** включает директорию **dir** в список поиска файлов, указанных в директиве *#include*.

### 2.2.8. -L

**Ключ -Ldir** добавляет директорию **dir** в список директорий, используемых для поиска подключенных библиотек в процессе сборки.

### 2.2.9. -I-

**Ключ -I-** служит разделителем типов директорий поиска файлов, указанных в директиве *#include*. Все директории, упомянутые до **-I-** используются для поиска файлов, указанных в *#include "имя файла"*, но не в *#include <имя файла>*. Директории, упомянутые после разделителя **-I-** могут использоваться для поиска любых *include*-файлов.

### 2.2.10. -Wa

**Ключ Wa,opt** передает **opt** в качестве опции ассемблеру. Если в опции **opt** встречаются запятые, она расщепляется запятыми на несколько опций.

### 2.2.11. -Wl

**Ключ -Wl,opt** передает **opt** в качестве опции линковщику. Если в опции **opt** встречаются запятые, она расщепляется запятыми на несколько опций.

### 2.2.12. -include

**Ключ -include file** указывает gcc компилировать сначала файл **file**, а потом уже основной файл. Это эквивалентно добавлению файла **file** в начало компилируемого. При этом любой ключ -D или -U из командной строки обрабатывается до **-include** независимо от порядка, в котором они записаны. Все ключи **-include** и -imacros обрабатываются в том порядке, в котором они записаны.

### 2.2.13. -imacros

**Ключ -imacros file** указывает gcc компилировать сначала файл **file**, а затем основной файл. При этом результат компиляции файла **file** отбрасывается. Это эквивалентно обработке и определению макросов.

Единственный эффект **-imacros file** состоит в том, что макроопределения, описанные в **file**, становятся доступны для применения в основном файле.

Любой ключ -D или -U из командной строки обрабатывается до **-imacros**, независимо от порядка, в котором они записаны. Все ключи -include и **-imacros** обрабатываются в том порядке, в котором они записаны.

### 2.2.14. -idirafter

**Ключ -idirafter dir** добавляет директорию **dir** ко второму маршруту включения в проект файлов директивами вида **include**. Второй маршрут включения используется, когда *include*-файл не найден в директориях, указанных ключами -I (главный маршрут включения).

### 2.2.15. -iprefix

**Ключ -iprefix prf** определяет **prf** как префикс для ключа -iwithprefix.

### 2.2.16. -iwithprefix

**Ключ -iwithprefix dir** добавляет директорию ко второму маршруту включения. Второй маршрут включения используется, когда *include*-файл не найден в директориях, указанных ключами -I (главный маршрут включения). Путь к директории получается из префикса, определенного ключом -iprefix, и параметра **dir**.

### 2.2.17. -iwithprefixbefore

**Ключ -iwithprefixbefore dir** добавляет директорию к главному маршруту включения (**include**). Путь к директории получается из префикса, определенного ключом -iprefix, и параметра **dir**.

### 2.2.18. -nostdinc

**Ключ -nostdinc** указывает gcc не искать заголовочные файлы в стандартных системных директориях. Поиск производится только в директориях, определенных ключом -I.

### 2.2.19. -g

**Ключ -g** указывает gcc произвести компиляцию с добавлением информации для отладчика.

### 2.2.20. -O

**Ключ -O** указывает gcc произвести компиляцию с оптимизацией.

### 2.2.21. -save-temps

**Ключ -save-temps** указывает gcc сохранить промежуточные файлы. Файлы будут сохранены в текущей директории с именами, зависящими от имени исходного файла.

### 2.2.22. -M

**Ключ -M** указывает препроцессору выводить правила для *make*, описывающие зависимости каждого объектного файла. Для каждого исходного файла, препроцессор выводит одно *make*-правило. Это правило может быть одиночной строкой или может быть продолжено с помощью '\', если оно длинное. Список правил печатается в стандартный вывод.

Ключ **-M** предполагает наличие ключа -E.

Другой способ выводить *make*-правила - установить переменную окружения *DEPENDENCIES_OUTPUT*.

### 2.2.23. -MM

**Ключ -MM** подобен -M, но выводятся только заголовочные файлы пользователя, включенные с помощью *#include "файл"*. Системные заголовочные файлы, включенные с помощью *#include <файл>*, опускаются.

### 2.2.24. -MD

**Ключ -MD** подобен -M, но информация о зависимостях записывается в файлы с именами, получающимися при замене расширения с "*.c*" на "*.d*" в именах входных файлов. **-MD** не запрещает обычную компиляцию, в отличие от -M.

### 2.2.25. -MMD

**Ключ -MMD** подобен -MD, но выводятся только заголовочные файлы пользователя.

### 2.2.26. -MG

**Ключ -MG** указывает gcc обрабатывать отсутствующие заголовочные файлы как генерируемые файлы и считать, что они находятся в том же самом каталоге, что и исходный файл. Если вы используете **-MG**, вы также должны указать или -M, или -MM. -MG не поддерживается с -MD или -MMD.

### 2.2.27. -H

**Ключ -H** указывает gcc выводить имя каждого используемого заголовочного файла.

### 2.2.28. -pipe

**Ключ -pipe** указывает gcc использовать каналы вместо временных файлов для связи между различными стадиями компиляции.

### 2.2.29. -time

**Ключ -time** указывает gcc выводить время, затраченное на исполнение каждой составляющей (**cpp**, **as**, **ld**,...).

### 2.2.30. -B

**Ключ -Bprf** задает префикс **prf**, который указывает, где искать выполняемые, библиотечные и заголовочные файлы, а также файлы данных для самого компилятора.

Для каждой запускаемой программы компилятор сначала пытается использовать префикс **-B**, если он указан. Если такое имя не найдено, или ключ **-B** не указан, компилятор пытается использовать два стандартных префикса: '*/usr/lib/gcc/*' и '*/usr/local/lib/gcc-lib/*'. Если ни один из этих префиксов не дает результата, не модифицированное имя программы ищется с использованием директорий указанных в вашей переменной окружения *'PATH'*.

Префикс **-B**, который эффективно указывает имена директорий, также применяется к библиотекам в линковщике и заголовочным файлам в препроцессоре.

### 2.2.31. -V

**Ключ -V ver** указывает, что нужно запускать версию GNU CC **ver**. Если ключ не указан, по умолчанию запускается установленная Вами версия.

### 2.2.32. -mcpu

**Ключ -mcpu=CPU** указывает на тип ЦП, определенный в **CPU**. Тип ЦП может принимать следующие значения:

*'r2000'*, *'r3000'*, *'r3900'*, *'r4000'*, *'r4100'*, *'r4300'*, *'r4400'*, *'r4600'*, *'r4650'*, *'r5000'*, *'r6000'*, *'r8000'* и *'orion'*. При этом, *'r2000'*, *'r3000'*, *'r4000'*, *'r5000'*, и *'r6000'* могут быть сокращены как *'r2k'* (или *'r2K'*), *'r3k'* и т.д.

Тип ЦП планирует программу в соответствии с той или иной спецификой чипа. Без использования ключей -mipsX или -mabi компилятор не будет генерировать кода, который не соответствует уровню 1 MIPS ISA (набор инструкций архитектуры).

### 2.2.33. -mipsX

**Ключи -mips1**, **-mips2**, **-mips3**, **-mips4** и **-mips32** указывают использовать тот или иной набор инструкций:

- **Ключ -mips1** - набор инструкций MIPS ISA 1 уровня. На этом уровне по умолчанию используется *'r3000'*;
- **Ключ -mips2** - набор инструкций MIPS ISA 2 уровня (вероятный переход, инструкции квадратного корня). На этом уровне по умолчанию используется *'r6000'*;
- **Ключ -mips3** - набор инструкций MIPS ISA 3 уровня (64-битные инструкции). На этом уровне по умолчанию используются 'r4000' и 'r6000';
- **Ключ -mips4** - набор инструкций MIPS ISA 3 уровня (условный переход, prefetch, усовершенствованные FPU-команды). На этом уровне по умолчанию используются 'r6000' и 'r8000'.
- **Ключ -mips32** - набор инструкций MIPS32.

### 2.2.34. -mfp32

**Ключ -mfp32** предполагает доступность 32-х 32-разрядных регистров с плавающей точкой. Этот ключ установлен по умолчанию.

### 2.2.35. -mfp64

**Ключ -mfp64** предполагает доступность 32-х 64-разрядных регистров с плавающей точкой. Этот ключ устанавливается по умолчанию, если используется ключ -mips3.

### 2.2.36. -mgp32

**Ключ -mgp32** предполагает доступность 32-х 32-разрядных регистров общего назначения. Этот ключ установлен по умолчанию.

### 2.2.37. -mgp64

**Ключ -mgp64** предполагает доступность 32-х 64-разрядных регистров общего назначения. Этот ключ устанавливается по умолчанию, если используется ключ -mips3.

### 2.2.38. -mint64

**Ключ -mint64** устанавливает размеры типов **int** и **long** равными 64 битам. Установки по умолчанию описаны в ключе -mlong32.

### 2.2.39. -mlong64

**Ключ -mlong64** устанавливает размер типа **long** равным 64 битам. Установки по умолчанию описаны в ключе -mlong32.

### 2.2.40. -mlong32

**Ключ -mlong32** устанавливает размеры типа **long**, типа **int** и типа **указателя** равными 32 битам. Если ни один из ключей **-mlong32**, -mlong64 или -mint64 не установлен, то размер типов **int**, **long** и **указателей** зависит от выборов ABI и ISA. Для -mabi2 и -mabi=n32 типы **int** и **long** имеют длину 32 бита. Для -mabid тип **int** имеет длину 32 бита, а тип **long** - 64 бита. Для -mabi\kbi и -mips1 или -mips2, типы **int** и **long** имеют размер 32 бита. Для -mabi\kbi и более высоких ISA, длина типа **int** - 32 бита, типа **long** - 64 бита. Размер типов указателя – меньшая из длины типа **long** или длины регистров общего назначения (которая в свою очередь зависит от ISA).

### 2.2.41. -mabiX

**Ключи -mabiX (-mabi2, -mabi=o64, -mabi=n32, -mabid, -mabikbi)** указывают ABI, для которого следует генерировать код:

Для '32' заданный по умолчанию уровень - -mips1, для 'n32' - -mips3 и -mips4 в других случаях. Наоборот, с -mips1 или -mips2 значение по умолчанию ABI является '32'; в других случаях значение по умолчанию ABI - '64'.

### 2.2.42. -mmips-as

**Ключ -mmips-as** указывает генерировать код ассемблера MIPS с добавлением отладочной информации.

### 2.2.43. -mgas

**Ключ -mgas** указывает генерировать код GNU-ассемблера.

### 2.2.44. -msplit-addresses

**Ключ -msplit-addresses** позволяет генерировать код для отдельной загрузки старшей и младшей частей констант адреса. Это позволяет gcc не загружать старшие разряды адреса. Такая оптимизация требуется для GNU **as** и GNU **ld**. В некоторых встроенных системах, где GNU **as** и GNU **ld** являются стандартными, это значение установлено по умолчанию.

### 2.2.45. -mno-split-addresses

**Ключ -mno-split-addresses** указывает gcc не генерировать код для отдельной загрузки старшей и младшей частей констант адреса.

### 2.2.46. -mrnames

**Ключ -mrnames** указывает выводить код с использованием имен регистров программного обеспечения MIPS, вместо аппаратных имен (например, *A0* вместо *$4*). Единственный известный ассемблер, который поддерживает этот ключ - Algorithmics-ассемблер.

### 2.2.47. -mno-rnames

**Ключ -mno-rnames** указывает не выводить код с использованием имен регистров программного обеспечения MIPS, вместо аппаратных имен.

### 2.2.48. -mgpopt

**Ключ -mgpopt** указывает записать все данные, объявленные перед инструкциями, в секции данных, что позволяет ассемблеру MIPS генерировать ссылки на память размером в слово, вместо использования двух слов для коротких глобальных или статических элементов данных.

### 2.2.49. -mno-gpopt

**Ключ -mno-gpopt** указывает не записывать все данные, объявленные перед инструкциями, в секции данных.

### 2.2.50. -mstats

**Ключ -mstats** указывает gcc для каждой *non-inline* функции вставлять одну строку в стандартный файл ошибок для записи статистики о процессе компиляции (число сохраненных регистров, размер стека, и т.д.).

### 2.2.51. -mno-stats

**Ключ -mno-stats** указывает gcc для каждой *non-inline* функции не вставлять строку в стандартный файл ошибок для записи статистики о процессе компиляции (число сохраненных регистров, размер стека, и т.д.).

### 2.2.52. -mmemcpy

**Ключ -mmemcpy** указывает менять все блоковые перемещения памяти на вызовы соответствующих строковых функций ("*memcpy*" или "*bcopy*") вместо генерирования *inline*-кода.

### 2.2.53. -mno-memcpy

**Ключ -mno-memcpy** указывает не менять все блоковые перемещения памяти на вызовы соответствующих строковых функций ("*memcpy*" или "*bcopy*").

### 2.2.54. -mmips-tfile

**Ключ -mno-mips-tfile** указывает компилятору обрабатывать объектный файл **'file'** программой, до того как MIPS-ассемблер добавит поддержку отладки.

### 2.2.55. -mno-mips-tfile

**Ключ -mno-mips-tfile** указывает компилятору не обрабатывать объектный файл **'file'** программой, до того как MIPS-ассемблер добавит поддержку отладки. Если **-mno-mips-tfile** не установлен, никакие локальные переменные не будут доступны отладчику.

### 2.2.56. -msoft-float

**Ключ -msoft-float** указывает gcc генерировать код, содержащий вызовы библиотечных функций для работы с числами с плавающей точкой.

### 2.2.57. -mhard-float

**Ключ -mhard-float** указывает gcc генерировать код, содержащий инструкции для работы с числами с плавающей точкой. Ключ установлен по умолчанию при использовании не модифицированных источников.

### 2.2.58. -mabicalls

**Ключ -mabicalls** указывает генерировать псевдооперации *'.abicalls'*, *'.cpload'* и *'.cprestore'*.

### 2.2.59. -mnoabi-calls

**Ключ -mno-abi-calls** указывает генерировать псевдооперации *'.abicalls'*, *'.cpload'* и *'.cprestore'*.

### 2.2.60. -mlong-calls

**Ключ -mlong-calls** указывает gcc делать все вызовы *'JALR'* инструкцией, которая перед вызовом требует загрузить адрес функции в регистр. Этот ключ необходимо использовать при вызовах функций (не являющихся сквозными указателями) в сегментах памяти вне текущих 512 мегабайт.

### 2.2.61. -mno-long-calls

**Ключ -mno-long-calls** указывает gcc не делать вызовы *'JALR'* инструкцией, которая перед вызовом требует загрузить адрес функции в регистр.

### 2.2.62. -mhalf-pic

**Ключ -mhalf-pic** указывает поместить указатели на внешние ссылки в секцию данных, загрузить их и не помещать ссылки в секцию кода.

### 2.2.63. -mno-half-pic

**Ключ -mno-half-pic** указывает помещать ссылки на внешние указатели в секцию кода.

### 2.2.64. -membedded-pic

**Ключ -membedded-pic** указывает генерировать PIC-код, соответствующий некоторым встроенным системам. Все вызовы будут использовать относительный адрес ПК, все данные будут адресоваться с использованием *$gp* регистра. При этом глобальные переменные могут занимать не более 65536 байт. Это требование GNU **as** и GNU **ld**. В настоящее время это применяется на выходных файлах формата ECOFF и не применяется на выходных файлах формата ELF.

### 2.2.65. -mno-embedded-pic

**Ключ -mno-embedded-pic** указывает не генерировать PIC-код, соответствующий некоторым встроенным системам.

### 2.2.66. -membedded-data

**Ключ -membedded-data** указывает расположить, если возможно, переменные в секции данных только для чтения, если нет, то попробовать расположить переменные в компактной секции данных. Если и это не возможно, то переменные будут расположены в обычной секции данных. Это дает чуть более медленный код, чем по умолчанию, но при этом уменьшает количество оперативной памяти, требуемой при выполнении, что может быть предпочтительно для некоторых встроенных систем.

### 2.2.67. -mno-embedded-data

**Ключ -mno-embedded-data** указывает расположить переменные в обычной секции данных.

### 2.2.68. -msingle-float

**Ключ -msingle-float** сообщает gcc, что сопроцессор поддерживает операции над числами с плавающей точкой только с одинарной точностью.

### 2.2.69. -mdouble-float

**Ключ -mdouble-float** сообщает gcc, что сопроцессор поддерживает операции над числами с плавающей точкой с двойной точностью. Этот ключ установлен по умолчанию.

### 2.2.70. -m4650

**Ключ -m4650** устанавливает ключи -msingle-float и -mcpu=r4650.

### 2.2.71. -mips16

**Ключ -mips16** дает возможность использования 16-разрядных команд.

### 2.2.72. -mno-mips16

**Ключ -mno-mips16** запрещает использование 16-разрядных команд.

### 2.2.73. -mentry

**Ключ -mentry** позволяет использовать входные и выходные псевдоопции. Ключ может использоваться, только если установлен ключ -mips16.

### 2.2.74. -EL

**Ключ -EL** указывает компилировать код в режиме little endian. Предполагается, что необходимые библиотеки существуют.

### 2.2.75. -EB

**Ключ -EB** указывает компилировать код в режиме big endian. Предполагается, что необходимые библиотеки существуют.

### 2.2.76. -nocpp

**Ключ -nocpp** сообщает ассемблеру MIPS не выполнять препроцессор для пользовательских ассемблерных файлов (с расширением *'.s'*) при ассемблировании.

# 3. Ассемблер

## 3.1. Ассемблер

**Ассемблер (mipsel-elf32-as**, **as)** это программа для транслирования исходного кода в объектный файл. Код ассемблера RISC-ядра основан на gas проекта binutil GNU.

Запуск ассемблера осуществляется из командной строки. При этом задаются ключи и перечисляются имена входных файлов.

## 3.2. Ключи командной строки ассемблера

При запуске **ассемблера (as)** следует установить **ключи командной строки**. Ассемблер имеет следующие ключи:

- -a…;
- -D;
- --defsym;
- --em;
- -f ;
- --gstabs;
- --gdwarf2;
- --help;
- -I;
- -J ;
- -K;
- -L;
- --MD;
- -o;
- -R;
- --statistics;
- --strip-local-absolute;
- --traditional-format;
- --version;
- -W;
- --warn;
- --fatal-warnings;
- --itbl;
- -Z;
- --listing-lhs-width;
- --listing-lhs-width2;
- --listing-rhs-width;
- --listing-cont-lines.

Специальные ключи для MIPS конфигурации **GNU `as'**:

- -membedded-pic;
- -EB;
- -EL;
- -G;
- --trap;
- --break;
- -KPIC;
- -non_shared;
- -xgot;
- -32;

- -64;
- -mNNNN;
- -no-mNNNN;
- --construct-floats, --no-construct-floats.

Список ключей в дальнейшем может быть расширен. Пример запуска ассемблера с получением листинга:
*mas -al xxx.s*

### 3.2.1. -a...

**Ключ -a...** служит для указания параметров управления листингом:

- **c** - исключить области, которые относятся к отвергнутым при условном ассемблировании;
- **d** - пропустить опции отладки;
- **l** - добавить сгенерированный код;
- **m** - включить макрорасширения;
- **n** - опустить обработку форм;
- **s** - включить символы;
- **L** - включить статистику отладки;
- **=FILE** - вывести листинга в **FILE** (должен быть последним).

Параметры, следующие после **-a**, могут быть скомбинированы в один ключ. Например, **-aln**.

### 3.2.2. -D

**Ключ -D** включает механизм генерации отладочных сообщений.

### 3.2.3. --defsym

**Ключ --defsym SYM=VAL** устанавливает значение для символа **SYM**, указанное в **VAL** (должно быть константой).
Например, *--defsym A=10*.

### 3.2.4. --em

**Ключ --em=[mipslelf | mipsbelf | mipself]** эмулирует выход (**mipslelf** используется по умолчанию).

### 3.2.5. -f

**Ключ -f** позволяет пропустить обработку комментариев и пробелов.

### 3.2.6. --gstabs

**Ключ --gstabs** позволяет включить генерацию отладочной информации об использованиях директив .stab.

### 3.2.7. --gdwarf2

**Ключ --gdwarf2** включает генерацию отладочной информации **DWARF2**.

### 3.2.8. --help

**Ключ --help** выводит описание всех ключей ассемблера.

### 3.2.9. -I

**Ключ -I dir** добавляет директорию, указанную в **dir** в список поиска для директивы **.include**.Используйте этот ключ для добавления пути к списку каталогов, в которых as ищет файлы, указанные в директиве **.include**. Вы можете использовать опцию столько раз, сколько это нужно для включения различных путей. Поиск сначала всегда осуществляется в текущем рабочем каталоге, затем as ищет файлы в каталогах, заданных с помощью ключа командной строки **-I** в порядке их задания (слева направо).

### 3.2.10. -J

**Ключ -J** позволяет пользователю отключить предупреждения о переполнении.

### 3.2.11. -K

**Ключ -K** включает предупреждения, в случае изменения таблицы разностей для длинных смещений. As иногда изменяет код для директив типа **.word SYM1-SYM2**. Вы можете использовать ключ **-K** для выдачи предупреждений в таких случаях.

### 3.2.12. -L

**Ключ -L (--keep-locals)** позволяет сохранять локальные символы. Метки, начинающиеся с **'L'** (только верхний регистр) называются "локальными метками". Обычно Вы не видите таких меток при отладке, потому что они предназначены для использования программами типа компиляторов, которые создают ассемблерный код, а не вами. Обычно и as и **ld** опускают такие метки. Ключ указывает as оставлять эти **'L...'**-символы в объектном файле. Обычно, используя этот ключ, Вы также должны указать линковщику ld, чтобы он сохранял символы с именами, начинающимися на **'L'**.

### 3.2.13. --MD

**Ключ --MD ФАЙЛ** позволяет выводить в **ФАЙЛ** информацию о зависимостях.

### 3.2.14. -o

**Ключ -o OBJFILE** указывает имя результирующего объектного файла (**OBJFILE**). Если такой файл существует, он будет перезаписан. По умолчанию as считает результирующим файл *a.out*.

### 3.2.15. -R

**Ключ -R** позволяет положить секцию данных в секцию текста. Использование **-R** позволяет создавать более короткие адресные смещения.

### 3.2.16. --statistics

**Ключ --statistics** позволяет вывести различную статистику выполнения. Используется для вывода двух характеристик ресурсов, использованных as: максимальный размер занятого пространства во время ассемблирования (в байтах) и общее время ассемблирования (время, которое процессор исполнял as, в секундах).

### 3.2.17. --strip-local-absolute

**Ключ --strip-local-absolute** удаляет локальные абсолютные символы.

### 3.2.18. --traditional-format

**Ключ --traditional-format** указывает as использовать традиционный для платформы формат ассемблера.

### 3.2.19. --version

**Ключ --version** выводит версию ассемблера.

### 3.2.20. -W

**Ключ -W (--no-warn)** запрещает **as** выдавать предупреждения (warnings).

### 3.2.21. --warn

**Ключ --warn** разрешает as выдавать предупреждения (warnings).

### 3.2.22. --fatal-warnings

Ключ **--fatal-warnings** указывает **as** рассматривать все предупреждения (warnings) как ошибки.

### 3.2.23. --itbl

**Ключ --itbl INSTTBL** позволяет расширить набор инструкций инструкциями, соответствующими спецификациям, определенным в файле **INSTTBL**.

### 3.2.24. -Z

**Ключ -Z** указывает as, что объектный файл должен быть сгенерирован даже при наличии ошибок.

### 3.2.25. --listing-lhs-width

**Ключ --listing-lhs-width** устанавливает ширину колонки в словах для листинга.

### 3.2.26. --listing-lhs-width2

**Ключ --listing-lhs-width2** устанавливает ширину в словах линий продолжения.

### 3.2.27. --listing-rhs-width

**Ключ --listing-rhs-width** позволяет установить максимальную длину строки исходных файлов.

### 3.2.28. --listing-cont-lines

**Ключ --listing-cont-lines** позволяет установить максимальное число строк, используемых для вывода в листинге.

### 3.2.29. -membedded-pic

**Ключ -membedded-pic** указывает as сгенерировать **PIC**-код, соответствующий некоторым встроенным системам.

### 3.2.30. -EB

**Ключ -EB** позволяет использовать big-endian систему нумерации байтов, по которой младший байт расположен в старших разрядах слова данных.

### 3.2.31. -EL

**Ключ -EL** позволяет использовать little-endian систему нумерации байтов, по которой младший байт расположен в младших разрядах слова данных.

### 3.2.32. -G

**Ключ -G NUM** позволяет поместить глобальные и статические данные меньшие или равные **NUM** байтам в секцию компактных данных или в bss-секцию, вместо секции обычных данных или bss. Это позволяет ассемблеру использовать инструкции с однословными ссылками на память, основанные на глобальных указателях (GP или $28), вместо двухсловных ссылок. По умолчанию при использовании ассемблера MIPS, **NUM** = 8. При использовании GNU-ассемблера, **NUM** = 0.

### 3.2.33. --trap

**Ключ --trap (--no-break)** указывает as генерировать такой код, чтобы при обнаружении ошибки получать **trap**-исключение, а не **break**-исключение. **Trap**-инструкции поддерживаются только в ISA второго уровня и выше.

### 3.2.34. --break

**Ключ --break (--no-trap)** указывает **as** генерировать такой код, чтобы при обнаружении ошибки получать **break**-исключение, а не **trap**-исключение. Ключ используется по умолчанию.

### 3.2.35. --KPIC

**Ключ --KPIC (--call_shared)** указывает **as** генерировать SVR4 позиционно-независимый код.

### 3.2.36. -non_shared

**Ключ --non_shared** указывает as не генерировать позиционно-независимый код.

### 3.2.37. -xgot

**Ключ -xgot** устанавливает 32-битный **GOT**.

### 3.2.38. -32

**Ключ -32** указывает as создавать 32-разрядный объектный файл (по умолчанию).

### 3.2.39. -64

**Ключ -64** указывает as создавать 64-разрядный объектный файл (по умолчанию).

### 3.2.40. -mNNNN

**Ключ -mNNNN** указывает as генерировать код для чипа MIPS R**NNNN**.

### 3.2.41. -no-mNNNN

**Ключ -no-mNNNN** указывает **as** не генерировать кода для чипа MIPS R**NNNN**.

### 3.2.42. --construct-floats

**Ключ --construct-floats** указывает as создавать константы с плавающей точкой двойной точности при помощи загрузки двух половинок значения в два регистра одиночной точности, из которых собирается регистр двойной точности. Этот ключ установлен по умолчанию.

### 3.2.43. --no-construct-floats

**Ключ --no-construct-floats** не дает as создавать константы с плавающей точкой двойной точности при помощи загрузки двух половинок значения в два регистра одиночной точности, из которых собирается регистр двойной точности. Это свойство полезно, если процессор поддерживает FR-бит в своем статусном регистре, и программно доступен. FR-бит запрещает преобразование регистра одиночной точности в регистр двойной точности.

## 3.3. Работа ассемблера

Ассемблер последовательно обрабатывает все строки файла. При этом сначала выполняются все директивы и макроподстановки, а затем полученный результат ассемблируется. После обработки всего файла выполняется окончательная обработка выражений и те из них, которые не могут быть вычислены на этом этапе, остаются для компоновщика.

## 3.4. Формат исходного файла Ассемблера

Входными файлами **as** являются файлы, содержащие программу RISC, написанную на языке *Ассемблера*. Подробнее о принципах написания программы RISC на языке *Ассемблера* см. книгу "**MC Programmer's Guide**" раздел "**Программирование под RISC**".

## 3.5. Сообщения об ошибках и предупреждения

As может выдавать **предупреждения** (warnings) и **сообщения об ошибках** в стандартный файл ошибок (обычно, терминал). Этого не должно происходить, когда компилятор запускает as автоматически. **Предупреждения** делаются, предполагая, что as может ассемблировать дефектную программу, а **сообщения об ошибках** выдаются при серьезных проблемах, которые прекращают **ассемблирование**.

**Предупреждения** имеют следующий формат:
**имя_файла**:*NNN*:**Текст Предупреждения**
(где *NNN* - номер строки).

Если было задано имя логического файла ([**.line**]), то он используется для вычисления выводимого номера, иначе выводится текущая строка обрабатываемого исходного файла.

**Сообщения об ошибках** имеют формат:
**Имя_файла**:*NNN*:**FATAL**:**Текст Сообщения Об Ошибке**
Имя файла и номер строки определяются так же, как и для предупреждения.

Для того чтобы as обрабатывал **предупреждения** так же, как **сообщения об ошибках**, используется ключ командной строки **as**: --fatal-warnings.

## 3.6. Условное ассемблирование

**Условное ассемблирование** позволяет генерировать код в зависимости от каких-либо условий. В частности, этот механизм может быть использован для вложенных макроопределений. Пример:

```
 9                zzz  .macro  f
 10                  .dl 12-\f
 11                  .ifge  \f
 12                  zzz "(\f-1)"
 13                  .endif
 14                  .endm
 15                  zzz 1
 15 0002 0000000B    > .dl 12-1
 15                > .ifge 1
 15                > zzz "(1-1)"
 15 0003 0000000C    >> .dl 12-(1-1)
 15                >> .ifge (1-1)
 15                >> zzz "((1-1)-1)"
 15 0004 0000000D    >>> .dl 12-((1-1)-1)
 15                >>> .ifge ((1-1)-1)
 15                >>> zzz "(((1-1)-1)-1)"
 15                >>> .endif
 15                >> .endif
 15                > .endif
```

Данный пример был получен при компиляции с ключами -alm. Эта комбинация ключей позволяет полностью проверить процедуру макроопределения. Соответственно, символами ">" в листинге указан уровень вложенности макроса.

Имеются следующие директивы **условного ассемблирования**:

- .if условие — проверка на неравенство нулю
- **.ifeq** выражение — проверка на равенство нулю
- **.ifge** выражение — проверка на больше или равно нулю
- **.ifgt** выражение — проверка на больше нуля
- **.ifle** выражение — проверка на меньше или равно нулю
- **.iflt** выражение — проверка на меньше нуля
- **.ifne** выражение — проверка на неравенство нулю
- .ifdef имя — проверить определенность имени
- .ifndef имя — проверить неопределенность имени
- .ifnotdef имя — проверить неопределенность имени
- **.ifc** строка1,строка2 — проверить строки на совпадение
- **.ifnc** строка1,строка2 — проверить строки на несовпадение
- **.ifeqs** строка1,строка2 — проверить С-строки на совпадение
- **.ifnes** строка1,строка2 — проверить С-строки на несовпадение
- .else — часть "иначе"
- **.elseif** условие — альтернативное условие
- .endif — конец условия

Под С-строкой здесь понимается строка в кавычках ("").

## 3.7. Директивы ассемблера

Все **ассемблерные директивы** имеют имена, начинающиеся с точки (.). Остальная часть имени пишется буквами, обычно строчными.

В списке присутствуют те **директивы**, которые доступны независимо от целевой платформы, под которую сконфигурирован GNU-ассемблер. Некоторые машинные конфигурации поддерживают дополнительные директивы.

- .abort;
- .align;
- .ascii;
- .asciz;
- .balign;
- .byte;
- .comm;
- .data;
- .def;
- .desc;
- .eject;
- .else;
- .end;
- .endef;
- .endif;
- .equ;
- .ent;
- .extern;
- .file;
- .fill;
- .global;
- .hword;
- .ident;
- .if;
- .ifdef;
- .ifnotdef;
- .include;
- .int;
- .irp;
- .irpc;
- .lcomm;
- .line;
- .ln;
- .list;
- .long;
- .macro;
- .endm;
- .exitm;
- \@;
- .nolist;
- .octa;
- .p2align;
- .psize;
- .quad;
- .rept;
- .sbttl;
- .section;
- .set;
- .short;
- .size;
- .space;
- .stab;
- .string;
- .tag;
- .text;

- .title;
- .type;
- .val;
- .word.

### 3.7.1. .abort

**Директива .abort** немедленно останавливает **ассемблирование**.

### 3.7.2. .align

**Директива .align aX,aY** осуществляет выравнивание (в данной подсекции) до некоторой границы. Абсолютное выражение **aX** есть число байт, необходимых для выравнивания. Выражение **aY** (также абсолютное) - значение, которым следует заполнить эти байты. Это выражение, а также запятую можно пропустить.

Например, `.short 0` выделит место под 16-разрядное число. При этом, старшие 16 разрядов останутся неиспользованными и при попытке выделить в памяти место под 32-х разрядное число, половина числа попадет в эти старшие 16 разрядов. Для предотвращения подобного используется директива **.align**. В данном случае, `.align 4,0` заполнит нулями эти 16 разрядов.

### 3.7.3. .ascii

**Директива .ascii Str1,Str2,..,StrN** ассемблирует строки (без автоматической подстановки нулевого байта в конец строки) в последовательные адреса памяти. Строк может быть ноль и более, все строки вводятся согласно правилам ввода литерных констант и разделяются запятыми.

Например, ***.ascii*** *"abc","xyz"*.

### 3.7.4. .asciz

**Директива .asciz Str1,Str2,..,StrN** ассемблирует строки (с автоматической подстановкой нулевого байта в конец строки) в последовательные адреса памяти. Строк может быть ноль и более, все строки вводятся согласно правилам ввода литерных констант и разделяются запятыми.

Например, ***.asciz*** *"abc","xyz"*.

### 3.7.5. .balign

**Директива .balign aX,aY** расширяет счетчик места (в данной подсекции) до некоторой границы. Первое выражение (которое должно быть абсолютным) есть требуемое выравнивание. Например, **.balign** *8* увеличивает счетчик места до кратного 8. Если счетчик места уже кратен 8, то никаких изменений не нужно.

### 3.7.6. .byte

**Директива .byte X1,X2,..,XN** последовательно ассемблирует значения выражений X1..XN в память. На каждое значение выделяется один байт. Например, директива ***.byte*** *10,14,22* последовательно ассемблирует в память числа 10, 14 и 22.

### 3.7.7. .comm

**Директива .comm Symbol,aLength** объявляет поименованную как **Symbol** общую область в секции bss. Обычно ld во время линковки резервирует для этого адреса памяти так, что ни одна частичная программа не определяет положение символа. Используйте директиву **.comm** для того, чтобы указать ld, что размер этой области должен быть, по крайней мере, равен значению, указанному в **aLength** (в байтах). Ld выделяет пространство для каждого

**.comm**-символа длиной в столько байт, сколько указано в максимальном запросе всех слинкованных частичных программ. Длина **aLength** должна быть абсолютным выражением.

### 3.7.8. .data

**Директива .data aSection** указывает, что **as** должен ассемблировать последующие операторы в конец подсекции **data** с номером **aSection** (который является абсолютным выражением). Если номер подсекции опущен, то по умолчанию предполагается ноль.

### 3.7.9. .def

**Директива .def Symbol** указывает на начало определения отладочной информации для метки **Symbol**. Определение продолжается до директивы .endef.

### 3.7.10. .desc

**Директива .desc Symbol, aX** устанавливает дескриптор символического имени **Symbol** равным младшим 16 битам абсолютного выражения **aX**.

### 3.7.11. .eject

**Директива .eject** немедленно завершает текущую страницу листинга.

### 3.7.12. .else

**Директива .else** - часть поддержки условного ассемблирования в as. Эта директива означает начало секции кода для условного ассемблирования, если условие в предыдущем .if было ложным.

Подробнее об условном ассемблировании см. Условное ассемблирование.

### 3.7.13. .end

**Директива .end Symbol** заканчивает подпрограмму, открытую директивой .ent.

### 3.7.14. .endef

**Директива .endef** заканчивает определение символического имени, начатое с директивы .def.

### 3.7.15. .endif

**Директива .endif** - часть поддержки условного ассемблирования в as. Эта директива означает конец блока кода, ассемблируемого условно.

Подробнее об условном ассемблировании см. Условное ассемблирование.

### 3.7.16. .endr

**Директива .endr** определяет конец блока операторов, начинающегося с .irp, .irpc, или .rept.

### 3.7.17. .equ

**Директива .equ Symbol,X** устанавливает значение метки **Symbol** в выражение **X**. Директива аналогична директиве .set.

### 3.7.18. .ent

Директива **.ent Symbol** используется в паре с директивой .end для выделения начала и конца функции. Например:

```
  .ent  Func1
Func1:
 nop
 nop
 nop
 .end  Func1
```

Если при этом между директивами **.ent** и **.end** попадется директива **.ent** или **.end**, относящаяся к другой функции, компилятор выдаст ошибку.

***Примечание***: данные директивы не вносят никаких других особенностей в компиляцию кода. Использовать **.end** и **.ent** необязательно.

### 3.7.19. .extern

**Директива .extern** присутствует только для совместимости с другими ассемблерами. Ассемблером as она игнорируется, так как as рассматривает все неопределенные символические имена (метки) как внешние.

### 3.7.20. .file

**Директива .file Str** начинает новый логический файл. **Str** - строка с именем файла. Если необходимо задать пустое имя файла, следует поставить вместо **Str** две кавычки.

### 3.7.21. .fill

**Директива .fill aNumber, aSize, aX** заполняет память несколькими (**aNumber**) копиями байт размера **aSize**. Содержимое байт берется из 8-байтного числа, причем старшие 4 байта - нули, а младшие определены в **aX**. Порядок следования байт такой, как в компьютере, для которого производится ассемблирование. Размер **aSize** может быть больше или равен нулю, но если он больше 8, то он принимается за 8 для совместимости с другими ассемблерами. Значение **aNumber** может быть нулем, или больше нуля. Выражения **aNumber**, **aSize** и **aX** должны быть абсолютными. Если указано только выражение **aNumber**, то **aSize** полагается равным 1.

### 3.7.22. .global

**Директива .global Symbol** объявляет символическое имя (метку) **Symbol** глобальным, то есть видимым для ld. Если метка определяется в частичной программе, ее значение становится доступным для других частичных программ, слинкованных вместе с этой в одном модуле. В противном случае метка **Symbol** получит свои атрибуты из метки с тем же именем, но определенной в другом файле, слинкованном в эту же программу. Также существует директива **.globl Symbol**, эквивалентная директиве **.global**.

### 3.7.23. .hword

**Директива .hword X1,X2,..,XN** последовательно ассемблирует значения выражений **X1..XN** в память. На каждое значение выделяется два байта. Например, директива ***.hword*** *312,400* последовательно ассемблирует в память числа 312 и 400.

### 3.7.24. .ident

**Директива .ident** используется некоторыми ассемблерами для помещения меток в объектные файлы.

### 3.7.25. .if

**Директива .if aX** отмечает начало секции кода, которая является существенной частью исходной программы только в том случае, если абсолютное выражение **aX** не равно нулю. Конец условной части кода должен быть обозначен директивой .endif. Также можно включить код для обработки альтернативного случая (**aX** равно нулю), поставив директиву .else.

Подробнее об условном ассемблировании см. Условное ассемблирование.

### 3.7.26. .ifdef

**Директива .ifdef Symbol** указывает as ассемблировать следующий блок кода, только если символическое имя **Symbol** было определено. Блок должен заканчиваться директивой .endif.

Подробнее об условном ассемблировании см. Условное ассемблирование.

### 3.7.27. .ifnotdef

**Директива .ifnotdef Symbol** указывает as ассемблировать следующий блок кода, только если символическое имя **Symbol** не было определено. Блок должен заканчиваться директивой .endif. Также существует директива **.ifndef Symbol**, эквивалентная директиве **.ifnotdef**.

Подробнее об условном ассемблировании см. Условное ассемблирование.

### 3.7.28. .include

**Директива .include Str** обеспечивает включение вспомогательного файла с именем, указанным в **Str**, в текст исходной программы. Код из файла ассемблируется так, как будто он следует сразу за **.include**. Когда включенный файл кончается, продолжается ассемблирование исходного файла. Вы можете управлять путем поиска, используя ключ командной строки -I. Имя файла вводится как строка.

### 3.7.29. .int

**Директива .int X1,X2,..,XN** последовательно ассемблирует значения выражений **X1..XN** в память. Порядок следования байт и место, выделяемое под число, зависят от типа целевой машины.

### 3.7.30. .irp

**Директива .irp Parameter, X1, X2,..,XN** выполняет блок операторов **N** раз, последовательно придавая символу **Parameter** значения **X1..XN**. Блок операторов начинается с **.irp** и заканчивается директивой .endr. Для обращения к значению **Parameter** следует использовать **\Parameter**. Например, ассемблирование

```
.irp param,1,2,3
 move d\param, 10
.endr
```

ассемблируется как

```
move d1,10
move d2,10
move d3,10
```

Если после символа не указано никаких значений, блок операторов ассемблируется один раз с символом, установленным в нулевую строку.

#### 3.7.31. .irpc

**Директива .irpc Parameter, X** выполняет блок операторов **N** раз, последовательно придавая символу **Parameter** значения каждого знака **X**. Блок операторов начинается с **.irpc** и заканчивается директивой .endr. Для обращения к значению символу **Parameter** следует использовать **\Parameter**. Например, ассемблирование

```
.irp param,123
```

```
  move d\param, 10
 .endr
```
ассемблируется как
```
 move d1,10
 move d2,10
 move d3,10
```
Если после символа не указано никакого значения, блок операторов ассемблируется один раз с символом, установленным в нулевую строку.

### 3.7.32. .lcomm

**Директива .lcomm Symbol, aLength** резервирует количество байтов, определенное в **aLength** (абсолютное выражение), для локальной общей области, обозначенной меткой **Symbol**. Секция и значение метки получаются из этих байтов. Адреса выделяются в секции bss, так что в момент запуска программы являются нулевыми. Если символическое имя (метка) не объявлено директивой .global, оно не будет видимым для компоновщика ld.

### 3.7.33. .line

**Директива .line aNumber** меняет логический номер следующей за директивой строки на указанный в **aNumber**. Номер строки должен быть абсолютным выражением. Тем не менее, другой оператор на этой строке (после знака разделителя операторов) будет иметь номер, равный **aNumber**-*1*.

### 3.7.34. .ln

**Директива .ln aNumber** эквивалентна директиве .line.

### 3.7.35. .list

**Директива .list** управляет (вместе с директивой .nolist) созданием ассемблерных листингов. Эти две директивы управляют значением внутреннего счетчика (изначально обнуленного). Директива **.list** увеличивает его, а директива .nolist - уменьшает. Ассемблерный листинг создается всегда, когда счетчик больше нуля.

### 3.7.36. .long

**Директива .long X1,X2,..,XN** эквивалентна директиве .int.

### 3.7.37. .macro

**Директива .macro Name,p1,p2,..,pN** позволяет задавать макроопределение (макрос) с именем **Name** и аргументами **p1,..,pN** для использования в коде программы. Аргументы могут иметь или не иметь значения по умолчанию. При ассемблировании исходного кода вместо имени **Name** макроса, as подставляет блок операторов, задающих макроопределение. Этот блок находится между директивами **.macro** и .endm. Для обращения в блоке операторов макроопределения к значениям его аргументов, следует использовать запись **\pK**, где **pK** - нужный аргумент.

Например,

- **.macro** *comm* - задает макроопределение с именем *comm* без аргументов;
- **.macro** *theSum,A,B* - задает макроопределение *theSum* с аргументами *A* и *B*;
- **.macro** *theAnotherSum,A=5,B* - задает макроопределение с именем *theAnotherSum* и аргументами *A* и *B*, причем аргумент *A* имеет значение по умолчанию, равное *5*.

Для вызова макроопределения в коде программы достаточно просто ввести его имя и аргументы. Аргументы могут быть введены как позицией, так и ключевым словом. Например, вызовы *theSum,10,3* и *theSum,A=10,B=3* эквивалентны.

### 3.7.38. .endm

**Директива .endm** обозначает конец определения макроса, начатого последней директивой .macro.

### 3.7.39. .exitm

**Директива .exitm** используется для преждевременного выхода из макроса, начатого последней директивой .macro.

### 3.7.40. \@

**\@** - псевдопеременная, в которой as хранит число выполненных макросов. Вызов **\@** возможен только внутри определения макроса, то есть между директивами .macro и .endm.

### 3.7.41. .nolist

**Директива .nolist** управляет (вместе с директивой .list) созданием ассемблерных листингов. Эти две директивы управляют значением внутреннего счетчика (изначально обнуленного). Директива **.nolist** уменьшает его, а директива .list - увеличивает. Ассемблерный листинг создается всегда, когда счетчик больше нуля.

### 3.7.42. .octa

**Директива .octa X1,X2,..,XN** последовательно ассемблирует значения выражений **X1..XN** в память. На каждое значение выделяется 16 байт.

### 3.7.43. .p2align

**Директива .p2align aX, aY** выравнивает счетчик места (в данной подсекции) до некоторой границы. Первое абсолютное выражение **aX** есть требуемое минимальное число младших нулей в счетчике места после расширения. Например, **.p2align** *3* увеличивает счетчик места до кратного *8*.

Второе выражение **aY** (также абсолютное) задает значение, которое должно быть сохранено в добавляемых байтах. Если **aY** отсутствует, то добавляемые байты будут нулями.

### 3.7.44. .psize

**Директива .psize aRows,aColumns** используется для объявления количества строк (**aRows**) и колонок (**aColumns**), которые будут использованы для оформления страницы при создании листинга.

Если директива **.psize** не используется, то по умолчанию в листинге будет *60* строк и *200* колонок.

As создает новую страницу в любом случае, когда превышается число строк (или при использовании директивы .eject). Если **aRows**=*0*, то новые страницы будут создаваться только при помощи .eject.

### 3.7.45. .purgem

**Директива .purgem** позволяет удалить макроопределение.

### 3.7.46. .quad

**Директива .quad X1,X2,..,XN** последовательно ассемблирует значения выражений **X1..XN** в память. На каждое значение выделяется 8 байт. Если какое-либо значение не помещается в 8 байт, то выдается предупреждение (warning) и используются 8 младших байт.

### 3.7.47. .rept

**Директива .rept Number** повторяет последовательность операторов, заключенную между **.rept** и .endr, число раз, определенное в **Number**.

Например, ассемблирование

```
.rept   3
.long   0
.endr
```

эквивалентно ассемблированию

```
.long   0
.long   0
.long   0
```

### 3.7.48. .sbttl

**Директива .sbttl Title** устанавливает **Title** в качестве заглавия при создании ассемблерного листинга.

Эта директива влияет на последующие страницы так же, как и на текущую, если она появляется в первых десяти строках этой страницы.

### 3.7.49. .section

**Директива .section Name,Number** ассемблирует следующий за директивой код в конец подсекции с номером, определенным в **Number**, в COFF-секции с именем, указанном в **Name**. Если номер подсекции не указан, то as использует подсекцию номер ноль. Директива **.section text** эквивалентна директиве .text. Директива **.section data** эквивалентна директиве .data.

Директива **.section** введена только для поддержки произвольных имен секций. При выводе в *'a.out'*, например, это не допускается, даже с указанием стандартного для *'a.out'* имени секции в качестве параметра.

### 3.7.50. .set

**Директива .set Symbol,X** устанавливает значение символического имени **Symbol** в равным значению выражения **X**. При использовании этой директивы по отношению к глобальному символу, в объектный файл записывается последнее установленное значение.

### 3.7.51. .short

**Директива .short X1,X2,..,XN** последовательно ассемблирует значения выражений **X1..XN** в память. На каждое выражение выделяется половина слова памяти (16 бит).

### 3.7.52. .size

**Директива .size** создается компилятором для включения дополнительной информации для отладки в таблицу символов. Директива разрешена только между директивами **.def** и .endef.

### 3.7.53. .space

**Директива .space aNumber,aFill** последовательно записывает в память число байт, определенное в **aNumber** и заполненное значением **aFill**. Если значение **aFill** опущено, байты заполняются нулями. И **aNumber**, и **aFill** должны быть абсолютными выражениями.

### 3.7.54. .stab

**.stabd, .stabn, .stabs**

Эти три директивы формируют символы для использования их в символических отладчиках. Эти символы не входят в hash-таблицу as: на них не может быть каких-либо ссылок в исходном файле. Для символа указывается до пяти полей:

**STRING**

Это имя символа. Оно может содержать любые знаки кроме *\000*, так что это более общее понятие, чем обычные символические имена. Некоторые отладчики используют для кодирования произвольных сложных структур имена символов указанные в этом поле.

**TYPE**

Абсолютное выражение. Тип символа устанавливается в 8 младших бит этого выражения. Любые битовые маски разрешены, но ld и отладчики используют самые примитивные битовые маски.

**OTHER**

Абсолютное выражение. Этот атрибут символа устанавливается в младшие 8 бит выражения.

**DESC**

Абсолютное выражение. Дескриптор символа устанавливается в младшие 16 бит выражения.

**VALUE**

Абсолютное выражение, которое становится значением символа.

Если при чтении операторов **.stabd**, **.stabn** или **.stabs** выдано предупреждение (warning), то вероятно, что эти символы уже были созданы. Тогда в объектном файле получается наполовину сформированный символ.

Директивы совместимы с предыдущими ассемблерами.

**.stabd TYPE, OTHER, DESC**

Имя символа не пустая строка. Это null-указатель, для совместимости. Старые ассемблеры используют null-указатели, так что они расходуют место в объектном файле на пустые строки.

Значение символа установлено в счетчик места, и способно изменятся. Когда ваша программа слинкована, значение этого символа есть адрес счетчика места во время ассемблирования **.stabd**.

**.stabn TYPE, OTHER, DESC, VALUE**

Имя символа установлено в пустую строку.

**.stabs STRING, TYPE, OTHER, DESC, VALUE**

Все пять полей определены.

### 3.7.55. .string

**Директива .string Str** копирует знаки, указанные в строке **Str**, в объектный файл. Возможно задание нескольких строк, разделенных запятыми.

### 3.7.56. .tag

**Директива .tag Struct_Name** используется компиляторами для включения дополнительной информации в таблицу символов. Директива может существовать только между директивами .def и .endef. **Tag'и** используются для связи определений структур в таблице символов с элементами этих структур.

### 3.7.57. .text

**Директива .text aSection** указывает as, что он должен ассемблировать следующие за **.text** операторы в конец подсекции **text** с номером, определенным в **aSection** (абсолютное выражение). Если номер подсекции не указан, то **as** использует нулевую подсекцию.

### 3.7.58. .title

**Директива .title Header** устанавливает **Header** в качестве заглавия (вторая строчка после номера страницы и имени исходного файла) при создании ассемблерного листинга.

Эта директива влияет на следующие страницы так же, как и на текущую, если она появляется в первых десяти строках этой страницы.

### 3.7.59. .type

**Директива .type intValue** записывает **intValue** как атрибут типа элемента таблицы символов. Директива допустима только между директивами .def и .endef.

### 3.7.60. .val

**Директива .val Address** записывает **Address** как атрибут значения элемента таблицы символов. Директива допустима только между директивами .def и .endef.

### 3.7.61. .word

**Директива .word X1,X2,..,XN** последовательно ассемблирует значения выражений **X1..XN** в память. На каждое выражение выделяется одно слово памяти (32 бита).

## 3.8. Результат ассемблирования

В результате работы **ассемблера** выдается объектный/выполняемый код программы и листинг (по запросу).

# 4. Двоичные утилиты

## 4.1. Двоичные утилиты

В среде MCS имеются следующие двоичные утилиты:

- Mdump (**mipsel-elf32-objdump**) - программа проверки и получения разнообразной информации из объектного файла, в частности, дизассемблирование файла;
- Mcopy (**mipsel-elf32-objcopy**) - программа выполнения преобразования формата;
- Библиотекарь (**mipsel-elf32-ar**) - программа для создания библиотек объектных модулей;
- Nm (**mipsel-elf32-nm**) - программа для вывода таблиц символов.

## 4.2. Ключи программы mdump

Программа **mipsel-elf32-objdump** предназначена для проверки, анализа и обработки объектных и выполняемых файлов. **Mdump** включает в себя набор средств по отображению отдельных составляющих файлов, межформатному преобразованию, например, с генерацией S-records, дизассемблированию.

Дизассемблер предназначен для обратного преобразования объектного/выполняемого кода в код на языке **Ассемблера** с целью проверки и анализа его.

Замечание: Дизассемблер dump-памяти выводит в формате RISC-ядра, т.е. LSB.

Запуск программы **mdump** из командной строки: **mipsel-elf32-objdump** ключи файлы

Ключи программы **mdump**:

- **-a, --archive-headers** Показать информацию заголовка архива;
- **-f, --file-headers** Показать содержание общего заголовка файла;
- **-p, --private-headers** Показать содержимое объектного заголовка формата данных;
- **-h, --[section]-headers** Показать содержимое заголовков секций;
- **-x, --all-headers** Показать все заголовки;
- **-d, --disassemble** Показать содержимое исполняемых секций в ассемблерном виде;
- **-D, --disassemble-all** Показать содержимое всех секций в ассемблерном виде;
- **-S, --source** Показать код источника вместе с ассемблером;
- **-s, --full-contents** Показать полное содержимое всех секций;
- **-g, --debugging** Показать отладочную информацию в объектном файле;
- **-G, --stabs** Показать (в необработанной форме) STABS-информацию;
- **-t, --syms** Показать содержимое таблиц символов;
- **-T, --dynamic-syms** Показать содержимое динамической таблицы символов;
- **-r, --reloc** Показать таблицу перекрестных ссылок;
- **-R, --dynamic-reloc** Показать таблицу динамических перекрестных ссылок;
- **-V, --version** Показать версию программы;
- **-i, --info** Показать список поддерживаемых объектов и архитектур;
- **-H, --help** Показать описание ключей.

Следующие ключи необязательны:

- **-b, --target=BFDNAME** Определить объектный формат как *BFDNAME*;
- **-m, --architecture=MACHINE** Определить архитектуру как *MACHINE*;
- **-j, --section=NAME** Показать информацию только о секции *NAME*;
- **-M, --disassembler-options=OPT** Обработать текст *OPT*-дизассемблером;

- **-EB, --endian=big** Использовать формат *big endian* при дизассемблировании;
- **-EL, --endian=little** Использовать формат *little endian* при дизассемблировании;
- **--file-start-context** Брать контекст с начала файла (используется с ключом **-S**);
- **-l, --line-numbers** Включить номера строк и имена файлов в вывод;
- **-C, --demangle** Декодировать измененные\обработанные имена символов;
- **-w, --wide** Формат выводимых данных с более чем 80 столбцами;
- **-z, --disassemble-zeroes** Не пропускать блоки с нулями при дизассемблировании;
- **--start-address=ADDR** Обрабатывать только данные с адресом, большим или равным *ADDR*;
- **--stop-address=ADDR** Обрабатывать только данные с адресом, меньшим или равным *ADDR*;
- **--prefix-addresses** Выводить полный адрес вместе с дизассемблированным текстом;
- **--[no-]show-row-insn** Показывать рядом с символьным дизассемблированным кодом шестнадцатеричный код;
- **--adjust-vma=OFFSET** Добавить смещение *OFFSET* ко всем выводимым адресам секций.

**Objdump** поддерживает следующие архитектуры: **elf32-littlemips**, **elf32-bigmips**, **elf64-bigmips**, **elf64-littlemips**, **elf64-little**, **elf64-big**, **elf32-little**, **elf32-big**, **srec**, **symbolsrec**, **tekhex**, **binary**, **ihex**.

## 4.3. Ключи программы mcopy

Программа **mipsel-elf32-objcopy** (**mcopy**) предназначена для выполнения преобразований над объектным файлом. Это и преобразования формата файла, и преобразования таблицы имен, и т.п. Допустимые форматы входных файлов могут быть получены при запуске программы с ключом **--help**.

Запуск программы **mcopy** из командной строки: **mipsel-elf32-objcopy** <ключи> входной файл [выходной файл]

Ключи программы **mcopy**:

- **-I, --input-target <bfd-имя>** Принять входной файл в *bfd*-фомате;
- **-O, --output-target <bfd-имя>** Создать выходной файл в *bfd*-формате;
- **-F, --target <bfd-имя>** Установить в <*bfd-имя*> как входной, так и выходной файлы;
- **--debugging** Преобразовать отладочную информацию, если возможно;
- **-p, --preserve-dates** Копирование изменения/доступа временных отметок на выходе;
- **-j, --only-section <имя>** Копировать только секцию <*имя*> на выход;
- **-R, --remove-section <имя>** Удалить секцию <*имя*> из выхода;
- **-S, --strip-all** Удалить все символы и информацию перемещения;
- **-g, --strip-debug** Удалить все символы отладки;
- **--strip-unneeded** Удалить все символы, не требующиеся для перемещений;
- **-N, --strip-symbol <имя>** Не копировать символ <*имя*>;
- **-K, --keep-symbol <имя>** Копировать только символ <*имя*>;
- **-L, --localize-symbol <имя>** Пометить символ <*имя*> как локальный;

- **-W, --weaken-symbol <имя>** Пометить символ <*имя*> как *weak*;
- **--weaken** Пометить все глобальные символы как *weak*;
- **-x, --discard-all** Удалить все глобальные символы;
- **-X, --discard-locals** Удалить все созданные компилятором символы;
- **-I, --interleave <N>** Копировать только один из каждых <*N*> байт;
- **-b, --byte <N>** Выделить байт номер <*N*> в каждом вложенном блоке;
- **--gap-fill <X>** Заполнить промежутки между секциями величиной <*X*>;
- **--pad-to <Addr>** Заполнить последнюю секцию вплоть до адреса <*Addr*>;
- **--set-start <Addr>** Установить начальный адрес равным <*Addr*>;
- **{--change-start|--adjust-start} <incr>** Добавить <*incr*> к начальному адресу;
- **{--change-addresses|--adjust-vma} <incr>** Добавить <*incr*> к LMA, VMA и стартовым адресам;
- **{--change-section-address|--adjust-section-vma} <имя> {=|+|-} <X>** Изменить LMA и VMA секции <*имя*> на величину <*X*>;
- **--change-section-lma <имя>{=|+|-}<X>** Изменить LMA секции <*Name*> на величину <*X*>;
- **--change-section-vma <имя>{=|+|-}<X>** Изменить VMA секции <*Name*> на величину <*X*>;

**{--[no-]change-warnings|--[no-]adjust-warnings}** Предупредить, если названная секция не существует;

- **--set-section-flags <имя>=<флаги>** Установить свойства секции <*имя*> в соответствии с <*флаги*>;
- **--add-section <имя>=<файл>** Добавить секцию <*имя*>, найденную в <*файле*> в выход;
- **--remove-leading-char** Удалить первый символ из глобальных символов;
- **--redefine-sym <старый>=<новый>** Переопределить символ с именем <*старый*> на <*новый*>;
- **-v, --verbose** Перечислить все объектные файлы, которые были изменены;
- **-V, --version** Вывести номер версии программы;
- **--help** Вывести описание ключей программы.

**Mcopy** поддерживает следующие форматы: **elf32-elcore**, **elf32-little**, **elf32-big**, **srec**, **symbolsrec**, **tekhex**, **binary**, **ihex**.

## 4.4. Ключи библиотекаря

**Библиотекарь** (**mipsel-elf32-ar**) позволяет создавать библиотеки объектных модулей. **Библиотекарь** выполняет следующие функции:

- создание библиотеки модулей;
- добавление объектного файла в библиотеку;
- удаление и замена объектного файла в библиотеке.

Для ускорения выделения требуемых для сборки модулей из библиотеки **библиотекарь** создает в ней таблицу символов.

Запуск **библиотекаря** из командной строки: **mar** ключи имя_библиотеки имена_файлов

Ключи **библиотекаря**:

- **d** Удалить файлы из библиотеки;
- **m[ab]** Переместить файл(ы) в архив;
- **p** Вывести файлы, найденные в архиве;
- **q[f]** Быстрое (без выполнения всех действий над таблицей символов) добавление файла в архив;
- **r[ab][f][u]** Заменить файл в архиве или добавить файл в архив;
- **t** Выдать список всех файлов библиотеки;
- **x[o]** Извлечь файл из библиотеки.

Модификации команд **библиотекаря**:

- **[a]** Разместить файлы после указанного;
- **[b]** Разместить файлы до указанного;
- **[N]** Использовать *instance [count]* имени;
- **[f]** Отсечь имена вставленных файлов;
- **[P]** Использовать полные имена при сравнении;
- **[o]** Сохранить первоначальные даты;
- **[u]** Менять только файлы новее содержащихся в архиве.

Основные модификаторы **библиотекаря**:

- **[c]** Не предупреждать, если библиотека должна быть создана;
- **[s]** Создать указатель архива (*cf. ranlib*);
- **[S]** Не создавать таблицы символов;
- **[V]** Показать версию **библиотекаря**.

**Библиотекарь** поддерживает следующие архитектуры: **elf32-littlemips**, **elf32-bigmips**, **elf64-bigmips**, **elf64-littlemips**, **elf64-little**, **elf64-big**, **elf32-little**, **elf32-big**, **srec**, **symbolsrec**, **tekhex**, **binary**, **ihex**.

## 4.5. Nm

Программа **Nm** (**mipsel-elf32-nm**) предназначена для вывода таблицы символов.

Запуск **nm** из командной строки: **nm** [Ключ]... [FILE]...
Листинг таблицы символов берется из **[FILE]** (по умолчанию это *a.out*).

Ключи:

- **-a, --debug-syms** Показать таблицу символов только отладчика;
- **-A, --print-file-name** Писать имя файла перед таблицей символов;
- **-B** Тоже что и **--format=bsd**;
- **-C, --demangle** Декодировать таблицу символов низкого уровня в таблицу символов пользователя;
- **--no-demangle** Не искажать таблицу символов низкого уровня;
- **-D, --dynamic** Показать динамическую таблицу символов вместо нормальной таблицы символов;
- **--defined-only** Показать только определенные символы;
- **-f, --format=FORMAT** Использовать выходной формат *FORMAT*. *FORMAT* может быть *'bsd'*, *'sysv'* или *'posix'*. По умолчанию используется *'bsd'*;
- **-g, --extern-only** Показать только внешние символы;
- **-h, --help** Вывести информацию о ключах;
- **-l, --line-numbers** Использовать отладочную информацию, для того, что бы найти имя файла и номер строки для каждого символа;
- **-n, --numeric-sort** Сортировать таблицу символов по адресу;
- **-o** Тоже что и **-A**;

- **-p, --no-sort** Не сортировать таблицу символов;
- **-P, --portability** Тоже что и **--format=posix**;
- **-r, --reverse-sort** Обратный порядок сортировки;
- **-s, --print-armap** Включить указатель для символов из членов архива;
- **--size-sort** Сортировать символы по размеру;
- **-t, --radix=RADIX** Использовать *RADIX* для вывода величин символов;
- **--target=BFDNAME** Определить формат объекта как *BFDNAME*;
- **-u, --undefined-only** Показать только неопределенные символы;
- **-V, --version** Показать версию программы.

**Nm** поддерживает следующие архитектуры: **elf32-littlemips**, **elf32-bigmips**, **elf64-bigmips**, **elf64-littlemips**, **elf64-little**, **elf64-big**, **elf32-little**, **elf32-big**, **srec**, **symbolsrec**, **tekhex**, **binary**, **ihex**.

# 5. Редактор связей ld

## 5.1. Введение в mld

Компоновщик программ **mld** (**mipsel-elf32-ld**) осуществляет компоновку выполняемого файла из набора объектных файлов и, если это необходимо, библиотек.

Вызов компоновщика из командной строки: **mld** ключи файлы

## 5.2. Ключи командной строки компоновщика

**Компоновщик** mld (**mipsel-elf32-ld**) имеет следующие **ключи**:

- -A, --architecture;
- -b, --format;
- -c, --mri-script;
- -d, -dc, -dp;
- -e, --entry;
- -E, --export-dynamic;
- -EB;
- -EL;
- -g;
- -G, --gpsize;
- -l, --library;
- -L, --library-path;
- -m;
- -M, --print-map;
- -o, --output;
- -O;
- -r,-i, -relocateable;
- -s, --strip-all;
- -S, --strip-debug;
- -t, --trace;
- -T, --script;
- -u, --undefined;
- -v, --version;
- -V;
- -x, --discard-all;
- -X, --discard-locals;
- --discard-none;
- -y, --trace-symbol;
- -(, --start-group;
- -), --end-group;
- -Bdynamic, -dy, -call_shared;
- -Bstatic, -dn, -non_shared, -static;
- --check-sections;
- --no-check-sections;
- --cref;
- --defsym;
- --demangle;
- --gc-sections;
- --no-gc-sections;
- --help;
- -Map;

- --no-demangle;
- --no-keep-memory;
- --no-undefined;
- --noinhibit-exec;
- --oformat;
- --retain-symbols-file;
- -rpath;
- -rpath-link;
- -shared, -Bshareable;
- --sort-common;
- --split-by-file;
- --stats;
- --traditional-format;
- -Ttext;
- -Tdata;
- -Tbss;
- --verbose;
- --version-script;
- --warn-common;
- --warn-multiple-gp;
- --warn-once;
- --warn-section-align;
- --whole-archive;
- --wrap.

Формат всех файлов по умолчанию - ELF. Компоновщик mld поддерживает следующие форматы: **elf32-littlemips**, **elf32-bigmips**, **elf64-bigmips**, **elf64-littlemips**, **elf64-little**, **elf64-big**, **elf32-little**, **elf32-big**, **srec**, **symbolsrec**, **tekhex**, **binary**, **ihex**. Эмуляция по умолчанию - *elf32elmip*.

Если необходимо скомпоновать вместе объектные файлы RISC и DSP-ядер, то перед компоновкой объектные файлы DSP-ядра должны быть преобразованы специальной программой **elcopy**.

### 5.2.1. -A

**Ключ -A ARCH** (**--architecture ARCH**) указывает ld установить архитектуру **ARCH**.

### 5.2.2. -b

**Ключ -b TARGET** (**--format TARGET**) указывает ld определить формат входных файлов **TARGET**.

### 5.2.3. -c

**Ключ -c FILE** (**--mri-script FILE**) указывает ld прочитать скрипт **FILE** (в MRI-формате) компоновщика.

### 5.2.4. -d

**Ключи -d, -dc и -dp** указывают компоновщику сделать общие символы определенными. Эти три ключа эквивалентны. Несколько форм поддерживаются для обеспечения совместимости с другими линковщиками. Они позволяют отводить место для переменных, даже если формат выходного файла - переместимый.

### 5.2.5. -e

**Ключ -e ADDRESS** (**--entry ADDRESS**) устанавливает начальный адрес в **ADDRESS**.

### 5.2.6. -E

**Ключ -E** (**--export-dynamic**) экспортирует все динамические символы.

### 5.2.7. -EB

**Ключ -EB** указывает ld связать все объекты **big-endian**.

### 5.2.8. -EL

**Ключ -EL** указывает ld связать все объекты **little-endian**.

### 5.2.9. -g

**Ключ -g** игнорируется компоновщиком и поддерживается только для совместимости с другими утилитами.

### 5.2.10. -G

**Ключ -G SIZE** (**--gpsize SIZE**) устанавливает максимальный, определенный в **SIZE**, размер объектов для оптимизации с использованием регистра GP для формата объектного файла MIPS ECOFF.

### 5.2.11. -l

**Ключ -l LIBNAME** (**--library LIBNAME**) осуществляет поиск библиотеки **LIBNAME** и добавляет архивный файл с указанным именем в список файлов для линковки. Ключ может быть использован неограниченное количество раз.

### 5.2.12. -L

**Ключ -L DIRECTORY** (**--library-path DIRECTORY**) добавляет директорию поиска в список директорий, в которых ld будет искать архивные файлы (библиотеки) и управляющие скрипты ld. Вы можете использовать этот ключ неограниченное число раз. Директории просматриваются в том порядке, в котором они указываются в командной строке. Указанные директории просматриваются прежде директорий по умолчанию. Для всех файлов, указанных ключом -l, будет выполнен поиск во всех директориях, указанных ключом **-L**, независимо от порядка, в котором они находились в командной строке.

### 5.2.13. -m

**Ключ -m EMULATION** позволяет устанавливать эмуляцию **EMULATION** (только **elf32elmip**).

### 5.2.14. -M

**Ключ -M** (**--print-map**) выводит карту памяти - диагностическую информацию размещении символов линковщиком и о глобальных переменных.

### 5.2.15. -N

**Ключ -N** указывает установить секции текста и данных доступными для чтения и записи. Также указывает не выравнивать по странице сегмент данных.

### 5.2.16. -o

**Ключ -o FILE** (**--output FILE**) позволяет указать файл для записи результата. По умолчанию это файл *a.out*.

### 5.2.17. -O

**Ключ -O** включает оптимизацию выходного файла.

### 5.2.18. -r

**Ключи -r,-i и -relocateable** указывают ld создавать перемещаемый выходной файл, то есть файл, который впоследствии может быть использован в качестве входного файла ld. Обычно это называется частичной линковкой. В частности, в операционных системах, которые поддерживают стандартные магические номера UNIX-а, этот ключ также устанавливает магический номер выходного файла в OMAGIC. Если ключ не использовался, на выходе ld получится законченный файл.

### 5.2.19. -s

**Ключ -s** (**--strip-all**) указывает ld удалить всю информацию о символах.

### 5.2.20. -S

**Ключ -S** (**--strip-debug**) указывает удалить всю отладочную информацию.

### 5.2.21. -t

**Ключ -t** (**--trace**) указывает ld выводить имена всех входных файлов по мере их обработки.

### 5.2.22. -T

**Ключ -T FILE** (**--script FILE**) позволяет прочитать команды компоновщика из скрипта в файле **FILE**. Эти команды замещают скрипт ld, принятый по умолчанию, а не являются дополнением к нему, поэтому в файле должно быть определено все необходимое для описания целевого формата объектного файла.

### 5.2.23. -u

**Ключ -u SYMBOL** (**--undefined SYMBOL**) указывает ld начать с неопределенных ссылок на символы. Описывает символ **SYMBOL** как неопределенный. Это позволяет избежать проблем при использовании дополнительных модулей стандартных библиотек. Вы можете использовать ключ **-u** в командной строке несколько раз.

### 5.2.24. -v

**Ключ -v** (**--version**) выводит информацию о версии ld.

### 5.2.25. -V

**Ключ -V** выводит версию ld и информацию об эмуляции.

### 5.2.26. -x

**Ключ -x** (**--discard-all**) удаляет все локальные символы.

### 5.2.27. -X

**Ключ -X** (**--discard-locals**) удаляет все временные локальные символы. Для большинства систем это символы, имена которых начинаются с *'L'*. Этот ключ установлен по умолчанию.

### 5.2.28. --discard-none

**Ключ --discard-none** указывает не удалять локальные символы.

### 5.2.29. -y

**Ключ -y SYMBOL** (**--trace-symbol SYMBOL**) позволяет проследить упоминания символа **SYMBOL**, печатая имя каждого линкуемого файла, в котором этот символ появляется. Ключ может быть использован неограниченное количество раз. Это полезно, когда у Вас есть неопределенный символ, а Вы не знаете, где находится ссылка на него.

### 5.2.30. -(

**Ключ -(** (**--start-group**) указывает на начало группы библиотек. Элементами группы могут быть либо точные имена файлов, либо ключи -l. Указанные библиотеки многократно просматриваются, пока не остается ни одной новой неопределенной ссылки. Обычно архивы (библиотеки) просматриваются только один раз - в том порядке, в каком они были указаны в командной строке. Если символ в библиотеке требует ссылки на неопределенный символ, находящийся в библиотеке, указанной позднее в командной строке, то линковщик не сможет обработать эту ссылку. Группировка библиотек заставляет их все просматриваться многократно, пока все ссылки не будут обработаны. Использование этого ключа значительно замедляет работу линковщика, поэтому ее рекомендуется использовать только тогда, когда у Вас есть несколько библиотек, которые ссылаются друг на друга. Конец группы указывается ключом -).

### 5.2.31. -)

**Ключ -)** (**--end-group**) указывает на конец группы библиотек, начатой ключом -(. Элементами группы могут быть либо точные имена файлов, либо ключи -l. Указанные библиотеки многократно просматриваются, пока не остается ни одной новой неопределенной ссылки. Обычно архивы (библиотеки) просматриваются только один раз - в том порядке, в каком они были указаны в командной строке. Если символ в библиотеке требует ссылки на неопределенный символ, находящийся в библиотеке, указанной позднее в командной строке, то линковщик не сможет обработать эту ссылку. Группировка позволяет просматривать библиотеки до тех пор, пока все ссылки не будут обработаны. Использование этого ключа значительно замедляет работу линковщика, поэтому ее рекомендуется использовать только тогда, когда у Вас есть несколько библиотек, которые ссылаются друг на друга.

### 5.2.32. -BDynamic

**Ключи -BDynamic**, **-dy** и **-call_shared** позволяют использовать разделяемые библиотеки.

### 5.2.33. -BStatic

**Ключи -BStatic**, **-dn**, **-non_shared** и **-static** запрещают использование разделяемых библиотек.

### 5.2.34. --check-sections

**Ключ --check-sections** проверяет адреса секций на перекрытие. Ключ установлен по умолчанию.

### 5.2.35. --no-check-sections

**Ключ --no-check-sections** запрещает проверку секций на перекрытие.

### 5.2.36. --cref

**Ключ --cref** позволяет вывести таблицу перекрестных ссылок.

### 5.2.37. --defsym

**Ключ --defsym Symbol=Expression** определяет символ **Symbol** и присваивает ему значение выражения **Expression**.

### 5.2.38. --demangle

**Ключ --demangle** указывает изменять имена символов.

### 5.2.39. --gc-sections

**Ключ --gc-sections** указывает удалить неиспользуемые секции.

### 5.2.40. --no-gc-sections

**Ключ --no-gc-sections** указывает не удалять неиспользуемые секции. Ключ установлен по умолчанию.

### 5.2.41. --help

**Ключ --help** выводит информацию обо всех ключах.

### 5.2.42. -Map

**Ключ -Map FILE** указывает файл (**FILE**) для записи карты памяти, если установлен ключ -M.

### 5.2.43. --no-demangle

**Ключ --no-demangle** запрещает изменять имена символов.

### 5.2.44. --no-keep-memory

**Ключ --no-keep-memory** позволяет использовать меньше оперативной памяти и больше дискового пространства. Обычно, ld в целях оптимизации кэширует таблицы имен входных файлов в памяти. Этот ключ указывает ld не использовать данную оптимизацию, а считывать заново таблицы имен по необходимости. Ключ используется для того, чтобы избежать нехватки памяти при линковке очень больших файлов.

### 5.2.45. --no-undefined

**Ключ --no-undefined** запрещает неопределенные символы.

### 5.2.46. --noinhibit-exec

**Ключ --noinhibit-exec** позволяет сгенерировать выходной файл даже при наличии ошибок в процессе линковки.

#### 5.2.47. --oformat

**Ключ --oformat TARGET** определяет архитектуру выходного файла.

#### 5.2.48. --retain-symbols-file

**Ключ --retain-symbols-file FILE** указывает сохранить только символы, содержащиеся в файле **FILE**.

#### 5.2.49. -rpath

**Ключ -rpath PATH** устанавливает путь (**PATH**) поиска разделяемых библиотек во время выполнения.

#### 5.2.50. -rpath-link

**Ключ -rpath-link PATH** устанавливает путь (**PATH**) поиска разделяемых библиотек во время линковки.

#### 5.2.51. -shared

**Ключ -shared** (**-Bshareable**) позволяет создать разделяемую библиотеку.

#### 5.2.52. --sort-common

**Ключ --sort-common** указывает сортировать общие символы по размеру.

#### 5.2.53. --split-by-file

**Ключ --split-by-file** указывает разделить выходные секции для каждого файла.

#### 5.2.54. --stats

**Ключ --stats** позволяет вывести статистику использования памяти.

#### 5.2.55. --traditional-format

**Ключ --traditional-format** указывает ld использовать формат, установленный по умолчанию.

#### 5.2.56. -Ttext

**Ключ -Ttext ADDR** устанавливает адрес секции кода равным **ADDR**. По умолчанию адрес секции кода равен нулю.

#### 5.2.57. -Tdata

**Ключ -Tdata ADDR** устанавливает адрес секции данных равным **ADDR**. По умолчанию адрес секции данных равен нулю.

#### 5.2.58. -Tbss

**Ключ -Tbss ADDR** устанавливает адрес секции неинициализированных данных, введенных директивами .comm/.lcomm, равным **ADDR**.

#### 5.2.59. --verbose

**Ключ --verbose** позволяет выводить дополнительную информацию во время сборки.

#### 5.2.60. --version-script

**Ключ --version-script FILE** позволяет прочитать информационный скрипт **FILE** о версии программы.

#### 5.2.61. --warn-common

**Ключ --warn-common** указывает ld предупреждать, когда общий символ комбинируется с другим общим символом или с определением символа. Линковщики UNIX-а позволяют эту немного избыточную практику, но на других платформах линковщики иногда не разрешают совершать эту операцию. Этот ключ позволяет Вам найти потенциальную проблему, возникающую при объединении глобальных символов. К сожалению, некоторые библиотеки Си используют эту практику, так что Вы можете получить предупреждение как для символов в библиотеках, так и в своих программах.

#### 5.2.62. --warn-multiple-gp

**Ключ --warn-multiple-gp** указывает ld предупреждать, если многократно используются GP-величины.

#### 5.2.63. --warn-once

**Ключ --warn-once** указывает ld выдавать предупреждение только один раз на каждый неопределенный символ.

#### 5.2.64. --warn-section-align

**Ключ --warn-section-align** указывает ld предупреждать, если адрес секции меняется из-за выравнивания.

#### 5.2.65. --whole-archive

**Ключ --whole-archive** указывает взять все объекты из архивов.

#### 5.2.66. --wrap

**Ключ --wrap SYMBOL** позволяет использовать функции упаковки для символа **SYMBOL**.

### 5.3. Управляющий файл компоновщика

Для более точного задания правил компоновки следует использовать **управляющий файл** компоновщика. Комментарии в этом файле можно оформлять в C-стиле, т.е. при помощи */* … */*.

Правила компоновки задаются в **управляющем файле** директивами компоновщика.

Для задания выражений в **управляющем файле** можно использовать любые C-выражения, т.е. **=**, **+=**, **-=**, ***=**, **/=**, **<<=**, **>>=**, **&=**, **|=**. При этом "**.**" обозначает текущую позицию. После каждого выражения обязательно ставить "**;**".

Указание **стартовой точки**.
По умолчанию **стартовой точкой** является один из символов следующего списка:

- символ, заданный ключом -e компоновщика;
- символ, указанный в директиве компоновщика **ENTRY**;
- символ **start**, если он определен;
- первый байт **text**-секции, если он определен;
- адрес 0.

По умолчанию, среда MCS использует управляющий файл *<имя модуля>.xl*, например *dsp.xl*. Этот файл генерируется средой автоматически, при сборке проекта рекомендуется использовать именно этот файл. Чтобы использовать другой файл, следует в командной строке компоновщика заменить *%Unit.xl* на имя необходимого файла. Для получения более подробной информации о настройке инструментария в среде MCS см. книгу **MC Studio**.

## 5.4. Директивы компоновщика

**Компоновщик mld** имеет следующие директивы:

- SECTIONS;
- INCLUDE;
- FILE;
- GROUP;
- OUTPUT;
- SEARCH_DIR;
- STARTUP;
- OUTPUT_FORMAT;
- TARGET;
- ASSERT;
- EXTERN;
- NOCROSSREFS;
- OUTPUT_ARCH;
- PROVIDE;
- MEMORY.

**Директивы компоновщика** используются в управляющем файле компоновщика.

### 5.4.1. SECTIONS

Управляющий файл компоновщика должен содержать, как минимум, директиву **SECTIONS**. Директива **SECTIONS** задает расположение секций в памяти. В данной директиве задается группа присваиваний переменным и управление выводом секций. Например,

```
SECTIONS
{
    . = 0x100;
    .text : { *(.text) }
    . = 0x800;
    .data : { *(.data) }
    .bss : { *(.bss) }
}
```

В этом примере выполняются присваивания текущей позиции *"."* и вывод всех (*"*"*) секций *.text* с позиции *0x100* памяти, а остальных секций - с позиции *0x800*.
Полный синтаксис директивы следующий:

```
SECTION [ADDRESS] [(TYPE)] : [AT(LMA)]
{
  OUTPUT-SECTION-COMMAND
  OUTPUT-SECTION-COMMAND
} [>REGION] [AT>LMA_REGION] [:PHDR :PHDR ...] [=FILLEXP]
```

Здесь:

- **ADDRESS** - задает адрес для данной секции. Если адрес не указан, то компоновщик базируется либо на текущем значении счетчика, либо использует значения соответствующего **REGION**. Это виртуальный адрес;
- **TYPE** - задает тип секции:

- **NOLAD** - не загружать в память;
- **DSECT**, **COPY**, **INFO**, **OVERLAY** - не выделять места;
- **LMA** - задает адрес загрузки (по умолчанию совпадает с **ADDRESS**);
- **REGION** - указывает область для размещения;
- **LMA_REGION** - задает область для загрузки;
- **PHDR** - размещение в рамках сегментов;
- **FILLEXP** -задает значение для заполнения не специфицированных областей.

### 5.4.2. INCLUDE

**Директива INCLUDE** позволяет включить в текущей точке еще один управляющий файл компоновщика.

### 5.4.3. FILE

**Директива FILE (file1 file2 ... fileN)** (или **FILE(file1,file2,..,fileN)**) позволяет явно подключить дополнительные файлы для сборки.

### 5.4.4. GROUP

**Директива GROUP** аналогична директиве FILE, но предназначена только для библиотек. Директива означает необходимость многократного просмотра указанных библиотек до полного исчерпания возможностей разрешения имен.

### 5.4.5. OUTPUT

Директива **OUTPUT(NAME)** задает имя файла (**NAME**) для вывода.

### 5.4.6. SEARCH_DIR

**Директива SEARCH_DIR(dir1 dir2 ... dirN)** задает список директорий для поиска библиотек.

### 5.4.7. STARTUP

**Директива STARTUP(file)** аналогична директиве INCLUDE, но дополнительно требует размещения файла **file** первым.

### 5.4.8. OUTPUT_FORMAT

**Директива OUTPUT_FORMAT(format)** задает формат выходного файла.

### 5.4.9. TARGET

**Директива TARGET** задает формат входного файла.

### 5.4.10. ASSERT

**Директива ASSERT(Expression, Message)** проверяет корректность выражения **Expression** (например, ограниченность размера секции) и выводит диагностическое сообщение **Message**.

### 5.4.11. EXTERN

**Директива EXTERN** позволяет принудительно объявить набор имен неопределенным.

### 5.4.12. NOCROSSREFS

**Директива NOCROSSREFS** позволяет задать список секций (как правило, оверлейных), которым запрещено ссылаться друг на друга.

### 5.4.13. OUTPUT_ARCH

**Директива OUTPUT_ARCH** позволяет получить выходной файл в "чужой" архитектуре.

### 5.4.14. PROVIDE

**Директива PROVIDE(name=expression)** позволяет присвоить имени **name** значение **expression**. Присвоение осуществляется только в том случае, если имя **name** больше нигде не определено.

### 5.4.15. MEMORY

**Директива MEMORY** позволяет задать общую область памяти для размещения каких-либо секций.
Директива имеет следующий синтаксис:

```
MEMORY
{
  NAME[ATTR]:ORIGIN=ORIGIN, LENGTH = LEN
  ...
}
```

# 6. Предметный указатель

ld ....41
mcopy ....37
mdump ....36
mld ....41
Nm ....39
Асемблер для RISC ....19
Ассемблер для RISC
  Директивы ассемблера ....25
  Ключи командной строки ассемблера ....19
  Работа ассемблера ....24
  Результат ассемблирования ....35
  Сообщения об ошибках и предупреждения ....24
  Условное ассемблирование ....25
  Формат исходного файла Ассемблера ....24
Библиотекарь ....38
Введение в инструменты RISC ....8
Двоичные утилиты ....36
Дизассемблер и другие утилиты для работы с объектными файлами ....36
Директивы ассемблера ....25
  .abort ....27
  .align ....27
  .ascii ....27
  .asciz ....27
  .balign ....27
  .byte ....27
  .comm ....27
  .data ....28
  .def ....28
  .desc ....28
  .eject ....28
  .else ....28
  .end ....28
  .enddef ....28
  .endif ....28
  .endm ....32
  .endr ....28
  .ent ....29
  .equ ....28
  .exitm ....32
  .extern ....29
  .file ....29
  .fill ....29
  .global ....29
  .hword ....29
  .ident ....29
  .if ....30
  .ifdef ....30
  .ifnotdef ....30
  .include ....30
  .int ....30
  .irp ....30
  .irpc ....30
  .lcomm ....31
  .line ....31
  .list ....31
  .ln ....31
  .long ....31
  .macro ....31
  .nolist ....32
  .octa ....32
  .p2align ....32
  .psize ....32
  .purgem ....32
  .quad ....32
  .rept ....33
  .sbttl ....33
  .section ....33
  .set ....33
  .short ....33
  .size ....33
  .space ....33
  .stabd ....34
  .string ....34
  .tag ....34
  .text ....35
  .title ....35
  .type ....35
  .val ....35
  .word ....35
Директивы компоновщика ....49
  ASSERT ....50
  EXTERN ....50
  FILE ....50
  GROUP ....50
  INCLUDE ....50
  MEMORY ....51
  NOCROSSREFS ....51
  OUTPUT ....50
  OUTPUT_ARCH ....51
  OUTPUT_FORMAT ....50
  PROVIDE ....51
  SEARCH_DIR ....50
  SECTIONS ....49
  STARTUP ....50
  TARGET ....50
Инструменты RISC ....8
Ключи библиотекаря ....38
Ключи командной строки ассемблера ....19
  -32 ....23
  -64 ....23
  --a... ....20
  --break ....23
  --construct-float ....24
  -D ....20
  --defsym ....20
  -EB ....23
  -EL ....23
  --em ....20
  -f ....20
  --fatal-warnings ....22
  -G ....23
  --gdwarf2 ....20
  --gstabs ....20
  --help ....20
  -I ....21
  --itbl ....22
  -J ....21
  -K ....21
  --KPIC ....23
  -L ....21
  --listing-cont-lines ....22
  --listing-lhs-width ....22
  --listing-lhs-width2 ....22
  --listing-rhs-width ....22
  --MD ....21
  -membedded-pic ....22
  -mNNNN ....23
  --no-construct-float ....24
  -no-mNNNN ....23
  -non_shared ....23
  -o ....21
  -R ....21
  --statistics ....21
  --strip-local-absolute ....21
  --traditional-format ....22

--trap ....23
--version ....22
-W ....22
--warn ....22
-xgot ....23
-Z ....22
Ключи командной строки компоновщика ....41
-( ....45
-) ....45
-A ....42
-b ....42
-BDynamic ....45
-BStatic ....45
-c ....42
--check-sections ....45
--cref ....46
-d ....42
--defsym ....46
--demangle ....46
--discard-none ....45
-e ....42
-E ....43
-EB ....43
-EL ....43
-g ....43
-G ....43
--gc-sections ....46
--help ....46
-I ....43
-L ....43
-m ....43
-M ....43
-Map ....46
--no-check-sections ....46
--no-demangle ....46
--no-gc-sections ....46
--noinhibite-exec ....46
--no-keep-memory ....46
--no-undefined ....46
-o ....44
-O ....44
--oformat ....47
-r ....44
--retain-symbols-file ....47
-rpath ....47
-rpath-link ....47
-s ....44
-S ....44
-shared ....47
--sort-common ....47
--split-by-file ....47
--stats ....47
-t ....44
-T ....44
-Tbss ....47
-Tdata ....47
--traditional-format ....47
-Ttext ....47
-u ....44
-v ....44
-V ....44
--vebose ....47
--version-script ....48
--warn-common ....48
--warn-multiple-gp ....48
--warn-once ....48
--warn-section-align ....48
--whole-archive ....48
--wrap ....48
-x ....44
-X ....45
-y ....45
Ключи программы mcopy ....37
Ключи программы mdump ....36
Ключи программы nm ....39
Ключи С-компилятора ....9
-B ....13
-c ....10
-D ....11
-E ....11
-EB ....18
-EL ....18
-g ....12
-H ....13
-I ....11
-I- ....11
-idirafter ....12
-imacros ....11
-include ....11
-iprefix ....12
-iwithprefix ....12
-iwithprefixbefore ....12
-L ....11
-M ....12
-m4650 ....17
-mabi=064 ....15
-mabi=n32 ....15
-mabi2 ....15
-mabicalls ....16
-mabid ....15
-mabikbi ....15
-mabiX ....15
-mcpu ....13
-MD ....13
-mdouble-float ....17
-membedded-data ....17
-membedded-pic ....17
-mentry ....18
-mfp32 ....14
-mfp64 ....14
-MG ....13
-mgas ....15
-mgp32 ....14
-mgp64 ....14
-mgpopt ....15
-mhalf-pic ....17
-mhard-float ....16
-mint64 ....14
-mips1 ....14
-mips16 ....17
-mips2 ....14
-mips3 ....14
-mips32 ....14
-mips4 ....14
-mipsX ....14
-mlong32 ....14
-mlong64 ....14
-mlong-calls ....16
-MM ....12
-MMD ....13
-mmemcpy ....16
-mmips-as ....15
-mmips-tfile ....16
-mnoabi-calls ....16
-mno-embedded-data ....17
-mno-embedded-pic ....17
-mno-gpopt ....15
-mno-half-pic ....17
-mno-long-calls ....16
-mno-memcpy ....16

-mno-mips16 ..... 17
-mno-mips-tfile ..... 16
-mno-rnames ..... 15
-mno-split-addresses ..... 15
-mno-stats ..... 16
-mrnames ..... 15
-msingle-float ..... 17
-msoft-float ..... 16
-msplit-addresses ..... 15
-mstats ..... 15
-nocpp ..... 18
-nostdinc ..... 12
-o ..... 10
-O ..... 12
-pipe ..... 13
-S ..... 10
-save-temps ..... 12
-time ..... 13
-U ..... 11
--V ..... 13
-Wa ..... 11
-Wl ..... 11
Компоновщик ..... 41
Директивы компоновщика ..... 49
Ключи командной строки ..... 41
Управляющий файл компоновщика ..... 48
Ошибки ..... 24
Предупреждения ..... 24
Работа ассемблера ..... 24
Редактор связей ld ..... 41
Сообщения об ошибках и предупреждениях ..... 24
Управляющий файл компоновщика ..... 48
Условное ассемблирование ..... 25
Формат исходного файла ассемблера ..... 24