libyasm

Generated by Doxygen 1.8.3.1

Sat Jan 31 2015 01:25:13

Contents

1 Data Structure Index 1

1.1 Data Structures . . . 1

2 File Index 2

2.1 File List . . . 2

3 Data Structure Documentation 4

3.1 yasm_arch_machine Struct Reference . . . 4

3.1.1 Detailed Description . . . 4

3.1.2 Field Documentation . . . 4

3.2 yasm_arch_module Struct Reference . . . 4

3.2.1 Detailed Description . . . 6

3.2.2 Field Documentation . . . 6

3.3 yasm_assoc_data_callback Struct Reference . . . 9

3.3.1 Detailed Description . . . 10

3.3.2 Field Documentation . . . 10

3.4 yasm_bytecode Struct Reference . . . 10

3.4.1 Detailed Description . . . 11

3.4.2 Member Function Documentation . . . 11

3.4.3 Field Documentation . . . 11

3.5 yasm_bytecode_callback Struct Reference . . . 12

3.5.1 Detailed Description . . . 13

3.5.2 Member Enumeration Documentation . . . 13

3.5.3 Field Documentation . . . 14

3.6 yasm_dbgfmt_module Struct Reference . . . 16

3.6.1 Detailed Description . . . 17

3.6.2 Field Documentation . . . 17

3.7 yasm_dir_help Struct Reference . . . 18

3.7.1 Detailed Description . . . 18

3.7.2 Field Documentation . . . 18

3.8 yasm_directive Struct Reference . . . 19

3.8.1 Detailed Description . . . 19

3.8.2 Member Enumeration Documentation . . . 19

3.8.3 Field Documentation . . . 20

3.9 yasm_expr Struct Reference . . . 20

3.9.1 Detailed Description . . . 21

3.9.2 Field Documentation . . . 21
3.10 yasm_expr__item Struct Reference . . . 21
3.10.1 Detailed Description . . . 22
3.11 yasm_listfmt_module Struct Reference . . . 22
3.11.1 Detailed Description . . . 22
3.11.2 Field Documentation . . . 22
3.12 yasm_object Struct Reference . . . 23
3.12.1 Detailed Description . . . 24
3.12.2 Member Function Documentation . . . 24
3.12.3 Field Documentation . . . 24
3.13 yasm_objfmt_module Struct Reference . . . 25
3.13.1 Detailed Description . . . 25
3.13.2 Field Documentation . . . 26
3.14 yasm_overrides Struct Reference . . . 28
3.14.1 Detailed Description . . . 28
3.15 yasm_valparam::yasm_param Union Reference . . . 28
3.15.1 Detailed Description . . . 28
3.16 yasm_parser_module Struct Reference . . . 29
3.16.1 Detailed Description . . . 29
3.16.2 Field Documentation . . . 29
3.17 yasm_preproc_module Struct Reference . . . 30
3.17.1 Detailed Description . . . 31
3.17.2 Field Documentation . . . 31
3.18 yasm_reloc Struct Reference . . . 32
3.18.1 Detailed Description . . . 33
3.19 yasm_scanner Struct Reference . . . 33
3.19.1 Detailed Description . . . 33
3.20 yasm_stdmac Struct Reference . . . 34
3.20.1 Detailed Description . . . 34
3.20.2 Field Documentation . . . 34
3.21 yasm_valparam Struct Reference . . . 34
3.21.1 Detailed Description . . . 35
3.21.2 Member Enumeration Documentation . . . 35
3.21.3 Field Documentation . . . 35
3.22 yasm_value Struct Reference . . . 36
3.22.1 Detailed Description . . . 36
3.22.2 Field Documentation . . . 37

4 File Documentation **38**

4.1 libyasm.h File Reference . . . 38
4.1.1 Detailed Description . . . 39
4.2 libyasm/arch.h File Reference . . . 39
4.2.1 Detailed Description . . . 41
4.2.2 Typedef Documentation . . . 41
4.2.3 Enumeration Type Documentation . . . 42
4.2.4 Function Documentation . . . 43
4.3 libyasm/assocdat.h File Reference . . . 48
4.3.1 Detailed Description . . . 49
4.3.2 Typedef Documentation . . . 49
4.3.3 Function Documentation . . . 49
4.4 libyasm/bytecode.h File Reference . . . 50
4.4.1 Detailed Description . . . 53
4.4.2 Macro Definition Documentation . . . 53
4.4.3 Typedef Documentation . . . 53
4.4.4 Function Documentation . . . 54
4.5 libyasm/coretype.h File Reference . . . 64
4.5.1 Detailed Description . . . 66
4.5.2 Typedef Documentation . . . 66
4.5.3 Enumeration Type Documentation . . . 72
4.5.4 Function Documentation . . . 73
4.5.5 Variable Documentation . . . 74
4.6 libyasm/dbgfmt.h File Reference . . . 75
4.6.1 Detailed Description . . . 76
4.6.2 Function Documentation . . . 76
4.7 libyasm/errwarn.h File Reference . . . 77
4.7.1 Detailed Description . . . 79
4.7.2 Macro Definition Documentation . . . 79
4.7.3 Typedef Documentation . . . 79
4.7.4 Enumeration Type Documentation . . . 80
4.7.5 Function Documentation . . . 81
4.7.6 Variable Documentation . . . 86
4.8 libyasm/expr.h File Reference . . . 87
4.8.1 Detailed Description . . . 89
4.8.2 Macro Definition Documentation . . . 89
4.8.3 Typedef Documentation . . . 90

4.8.4 Enumeration Type Documentation . . . 91
4.8.5 Function Documentation . . . 91
4.9 libyasm/file.h File Reference . . . 98
4.9.1 Detailed Description . . . 100
4.9.2 Macro Definition Documentation . . . 100
4.9.3 Typedef Documentation . . . 109
4.9.4 Function Documentation . . . 109
4.10 libyasm/floatnum.h File Reference . . . 114
4.10.1 Detailed Description . . . 114
4.10.2 Function Documentation . . . 114
4.11 libyasm/hamt.h File Reference . . . 117
4.11.1 Detailed Description . . . 117
4.11.2 Typedef Documentation . . . 118
4.11.3 Function Documentation . . . 118
4.12 libyasm/intnum.h File Reference . . . 120
4.12.1 Detailed Description . . . 122
4.12.2 Function Documentation . . . 122
4.13 libyasm/linemap.h File Reference . . . 130
4.13.1 Detailed Description . . . 131
4.13.2 Function Documentation . . . 131
4.14 libyasm/listfmt.h File Reference . . . 134
4.14.1 Detailed Description . . . 134
4.14.2 Typedef Documentation . . . 135
4.14.3 Function Documentation . . . 135
4.15 libyasm/objfmt.h File Reference . . . 136
4.15.1 Detailed Description . . . 136
4.15.2 Function Documentation . . . 136
4.16 libyasm/parser.h File Reference . . . 138
4.16.1 Detailed Description . . . 139
4.16.2 Typedef Documentation . . . 139
4.17 libyasm/preproc.h File Reference . . . 139
4.17.1 Detailed Description . . . 140
4.17.2 Typedef Documentation . . . 140
4.17.3 Function Documentation . . . 140
4.18 libyasm/section.h File Reference . . . 142
4.18.1 Detailed Description . . . 144
4.18.2 Typedef Documentation . . . 144

4.18.3 Function Documentation . . . 144
4.19 libyasm/symrec.h File Reference . . . 152
4.19.1 Detailed Description . . . 155
4.19.2 Typedef Documentation . . . 155
4.19.3 Enumeration Type Documentation . . . 156
4.19.4 Function Documentation . . . 156
4.20 libyasm/valparam.h File Reference . . . 166
4.20.1 Detailed Description . . . 168
4.20.2 Typedef Documentation . . . 168
4.20.3 Function Documentation . . . 168
4.21 libyasm/value.h File Reference . . . 175
4.21.1 Detailed Description . . . 176
4.21.2 Function Documentation . . . 176

Index **179**

1 Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

yasm_arch_machine
YASM machine subtype **4**

yasm_arch_module
YASM architecture module interface **4**

yasm_assoc_data_callback
YASM associated data callback structure **9**

yasm_bytecode
A bytecode **10**

yasm_bytecode_callback
Bytecode callback structure **12**

yasm_dbgfmt_module
Debug format module interface **16**

yasm_dir_help
Directive valparam parse helper structure **18**

yasm_directive
Directive list entry structure **19**

yasm_expr
Expression 20

yasm_expr__item
Expression item 21

yasm_listfmt_module
YASM list format module interface 22

yasm_object
An object 23

yasm_objfmt_module
Object format module interface 25

yasm_overrides
Structure of functions that can be overridden 28

yasm_valparam::yasm_param
Parameter value 28

yasm_parser_module
YASM parser module interface 29

yasm_preproc_module
YASM preprocesor module interface 30

yasm_reloc 32

yasm_scanner
Re2c scanner state 33

yasm_stdmac
Standard macro structure for modules that allows association of a set of standard macros with a parser/preprocessor combination 34

yasm_valparam
Value/parameter pair 34

yasm_value
A value 36

2 File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

libyasm.h
YASM library primary header file 38

libyasm/arch.h
YASM architecture interface 39

libyasm/assocdat.h
YASM associated data storage (libyasm internal use) 48

libyasm/bytecode.h
YASM bytecode interface 50

libyasm/coretype.h
YASM core types and utility functions 64

libyasm/dbgfmt.h
YASM debug format interface 75

libyasm/errwarn.h
YASM error and warning reporting interface 77

libyasm/expr.h
YASM expression interface 87

libyasm/file.h
YASM file helpers 98

libyasm/floatnum.h
YASM floating point (IEEE) interface 114

libyasm/hamt.h
Hash Array Mapped Trie (HAMT) functions 117

libyasm/intnum.h
YASM integer number interface 120

libyasm/linemap.h
YASM virtual line mapping interface 130

libyasm/listfmt.h
YASM list format interface 134

libyasm/module.h ??

libyasm/objfmt.h
YASM object format module interface 136

libyasm/parser.h
YASM parser module interface 138

libyasm/preproc.h
YASM preprocessor module interface 139

libyasm/section.h
YASM section interface 142

libyasm/symrec.h
YASM symbol table interface 152

libyasm/valparam.h
YASM value/parameter interface 166

libyasm/value.h
YASM value interface 175

3 Data Structure Documentation

3.1 yasm_arch_machine Struct Reference

YASM machine subtype.

`#include <arch.h>`

Data Fields

- const char * name
  One-line description of the machine.
- const char * keyword
  Keyword used to select machine.

3.1.1 Detailed Description

YASM machine subtype.

A number of different machine types may be associated with a single architecture. These may be specific CPU's, but the ABI used to interface with the architecture should be the primary differentiator between machines. Some object formats (ELF) use the machine to determine parameters within the generated output.

Definition at line 74 of file arch.h.

3.1.2 Field Documentation

3.1.2.1 const char* yasm_arch_machine::keyword

Keyword used to select machine.

Definition at line 79 of file arch.h.

3.1.2.2 const char* yasm_arch_machine::name

One-line description of the machine.

Definition at line 76 of file arch.h.

The documentation for this struct was generated from the following file:

- libyasm/arch.h

3.2 yasm_arch_module Struct Reference

YASM architecture module interface.

`#include <arch.h>`

Data Fields

- const char * name

  One-line description of the architecture.
- const char * keyword

  Keyword used to select architecture.
- const yasm_directive * directives

  NULL-terminated list of directives.
- yasm_arch *(* create)(const char *machine, const char *parser, yasm_arch_create_error *error)

  Create architecture.
- void(* destroy)(yasm_arch *arch)

  Module-level implementation of yasm_arch_destroy().
- const char *(* get_machine)(const yasm_arch *arch)

  Module-level implementation of yasm_arch_get_machine().
- unsigned int(* get_address_size)(const yasm_arch *arch)

  Module-level implementation of yasm_arch_get_address_size().
- int(* set_var)(yasm_arch *arch, const char *var, unsigned long val)

  Module-level implementation of yasm_arch_set_var().
- yasm_arch_insnprefix(* parse_check_insnprefix)(yasm_arch *arch, const char *id, size_t id_len, unsigned long line, yasm_bytecode **bc, uintptr_t *prefix)

  Module-level implementation of yasm_arch_parse_check_insnprefix().
- yasm_arch_regtmod(* parse_check_regtmod)(yasm_arch *arch, const char *id, size_t id_len, uintptr_t *data)

  Module-level implementation of yasm_arch_parse_check_regtmod().
- const unsigned char **(* get_fill)(const yasm_arch *arch)

  Module-level implementation of yasm_arch_get_fill().
- int(* floatnum_tobytes)(yasm_arch *arch, const yasm_floatnum *flt, unsigned char *buf, size_t destsize, size_t valsize, size_t shift, int warn)

  Module-level implementation of yasm_arch_floatnum_tobytes().
- int(* intnum_tobytes)(yasm_arch *arch, const yasm_intnum *intn, unsigned char *buf, size_t destsize, size_t valsize, int shift, const yasm_bytecode *bc, int warn)

  Module-level implementation of yasm_arch_intnum_tobytes().
- unsigned int(* get_reg_size)(yasm_arch *arch, uintptr_t reg)

  Module-level implementation of yasm_arch_get_reg_size().
- uintptr_t(* reggroup_get_reg)(yasm_arch *arch, uintptr_t reggroup, unsigned long regindex)

  Module-level implementation of yasm_arch_reggroup_get_reg().
- void(* reg_print)(yasm_arch *arch, uintptr_t reg, FILE *f)

  Module-level implementation of yasm_arch_reg_print().
- void(* segreg_print)(yasm_arch *arch, uintptr_t segreg, FILE *f)

  Module-level implementation of yasm_arch_segreg_print().
- yasm_effaddr *(* ea_create)(yasm_arch *arch, yasm_expr *e)

  Module-level implementation of yasm_arch_ea_create().
- void(* ea_destroy)(yasm_effaddr *ea)

  Module-level implementation of yasm_arch_ea_destroy().
- void(* ea_print)(const yasm_effaddr *ea, FILE *f, int indent_level)

  Module-level implementation of yasm_arch_ea_print().
- yasm_bytecode *(* create_empty_insn)(yasm_arch *arch, unsigned long line)

  Module-level implementation of yasm_arch_create_empty_insn().
- const yasm_arch_machine * machines

NULL-terminated list of machines for this architecture.

- const char * default_machine_keyword

  Default machine keyword.

- unsigned int wordsize

  Canonical "word" size in bits.

- unsigned int min_insn_len

  Worst case minimum instruction length in bytes.

3.2.1 Detailed Description

YASM architecture module interface.

Note

All "data" in parser-related functions (yasm_arch_parse_*) needs to start the parse initialized to 0 to make it okay for a parser-related function to use/check previously stored data to see if it's been called before on the same piece of data.

Definition at line 88 of file arch.h.

3.2.2 Field Documentation

3.2.2.1 yasm_arch*(* yasm_arch_module::create)(const char *machine, const char *parser,yasm_arch_create_error *error)

Create architecture.

Module-level implementation of yasm_arch_create(). Call yasm_arch_create() instead of calling this function.

Definition at line 106 of file arch.h.

3.2.2.2 yasm_bytecode*(* yasm_arch_module::create_empty_insn)(yasm_arch *arch, unsigned long line)

Module-level implementation of yasm_arch_create_empty_insn().

Call yasm_arch_create_empty_insn() instead of calling this function.

Definition at line 202 of file arch.h.

3.2.2.3 const char* yasm_arch_module::default_machine_keyword

Default machine keyword.

Call yasm_arch_get_machine() to get the active machine of a particular yasm_arch.

Definition at line 215 of file arch.h.

3.2.2.4 void(* yasm_arch_module::destroy)(yasm_arch *arch)

Module-level implementation of yasm_arch_destroy().

Call yasm_arch_destroy() instead of calling this function.

Definition at line 112 of file arch.h.

3.2.2.5 const yasm_directive* yasm_arch_module::directives

NULL-terminated list of directives.

NULL if none.

Definition at line 100 of file arch.h.

3.2.2.6 yasm_effaddr*(* yasm_arch_module::ea_create)(yasm_arch *arch,yasm_expr *e)

Module-level implementation of yasm_arch_ea_create().

Call yasm_arch_ea_create() instead of calling this function.

Definition at line 187 of file arch.h.

3.2.2.7 void(* yasm_arch_module::ea_destroy)(yasm_effaddr *ea)

Module-level implementation of yasm_arch_ea_destroy().

Call yasm_arch_ea_destroy() instead of calling this function.

Definition at line 192 of file arch.h.

3.2.2.8 void(* yasm_arch_module::ea_print)(const yasm_effaddr *ea, FILE *f, int indent_level)

Module-level implementation of yasm_arch_ea_print().

Call yasm_arch_ea_print() instead of calling this function.

Definition at line 197 of file arch.h.

3.2.2.9 int(* yasm_arch_module::floatnum_tobytes)(yasm_arch *arch, const yasm_floatnum *flt, unsigned char *buf, size_t destsize, size_t valsize, size_t shift, int warn)

Module-level implementation of yasm_arch_floatnum_tobytes().

Call yasm_arch_floatnum_tobytes() instead of calling this function.

Definition at line 151 of file arch.h.

3.2.2.10 unsigned int(* yasm_arch_module::get_address_size)(const yasm_arch *arch)

Module-level implementation of yasm_arch_get_address_size().

Call yasm_arch_get_address_size() instead of calling this function.

Definition at line 122 of file arch.h.

3.2.2.11 const unsigned char**(* yasm_arch_module::get_fill)(const yasm_arch *arch)

Module-level implementation of yasm_arch_get_fill().

Call yasm_arch_get_fill() instead of calling this function.

Definition at line 146 of file arch.h.

3.2.2.12 const char*(* yasm_arch_module::get_machine)(const yasm_arch *arch)

Module-level implementation of yasm_arch_get_machine().

Call yasm_arch_get_machine() instead of calling this function.

Definition at line 117 of file arch.h.

3.2.2.13 unsigned int(* yasm_arch_module::get_reg_size)(yasm_arch *arch, uintptr_t reg)

Module-level implementation of yasm_arch_get_reg_size().

Call yasm_arch_get_reg_size() instead of calling this function.

Definition at line 166 of file arch.h.

3.2.2.14 int(* yasm_arch_module::intnum_tobytes)(yasm_arch *arch, const yasm_intnum *intn, unsigned char *buf, size_t destsize, size_t valsize, int shift, const yasm_bytecode *bc, int warn)

Module-level implementation of yasm_arch_intnum_tobytes().

Call yasm_arch_intnum_tobytes() instead of calling this function.

Definition at line 158 of file arch.h.

3.2.2.15 const char* yasm_arch_module::keyword

Keyword used to select architecture.

Call yasm_arch_keyword() to get the keyword of a particular yasm_arch.

Definition at line 97 of file arch.h.

3.2.2.16 const yasm_arch_machine* yasm_arch_module::machines

NULL-terminated list of machines for this architecture.

Call yasm_arch_get_machine() to get the active machine of a particular yasm_arch.

Definition at line 209 of file arch.h.

3.2.2.17 unsigned int yasm_arch_module::min_insn_len

Worst case minimum instruction length in bytes.

Call yasm_arch_min_insn_len() to get the minimum instruction length of a particular yasm_arch.

Definition at line 227 of file arch.h.

3.2.2.18 const char* yasm_arch_module::name

One-line description of the architecture.

Call yasm_arch_name() to get the name of a particular yasm_arch.

Definition at line 92 of file arch.h.

3.2.2.19 yasm_arch_insnprefix(* yasm_arch_module::parse_check_insnprefix)(yasm_arch *arch, const char *id, size_t id_len, unsigned long line,yasm_bytecode **bc,uintptr_t *prefix)

Module-level implementation of yasm_arch_parse_check_insnprefix().

Call yasm_arch_parse_check_insnprefix() instead of calling this function.

Definition at line 133 of file arch.h.

3.2.2.20 yasm_arch_regtmod(* yasm_arch_module::parse_check_regtmod)(yasm_arch *arch, const char *id, size_t id_len,uintptr_t *data)

Module-level implementation of yasm_arch_parse_check_regtmod().

Call yasm_arch_parse_check_regtmod() instead of calling this function.

Definition at line 140 of file arch.h.

3.2.2.21 void(∗ yasm_arch_module::reg_print)(yasm_arch ∗arch, uintptr_t reg, FILE ∗f)

Module-level implementation of yasm_arch_reg_print().

Call yasm_arch_reg_print() instead of calling this function.

Definition at line 177 of file arch.h.

3.2.2.22 uintptr_t(∗ yasm_arch_module::reggroup_get_reg)(yasm_arch ∗arch, uintptr_t reggroup, unsigned long regindex)

Module-level implementation of yasm_arch_reggroup_get_reg().

Call yasm_arch_reggroup_get_reg() instead of calling this function.

Definition at line 171 of file arch.h.

3.2.2.23 void(∗ yasm_arch_module::segreg_print)(yasm_arch ∗arch, uintptr_t segreg, FILE ∗f)

Module-level implementation of yasm_arch_segreg_print().

Call yasm_arch_segreg_print() instead of calling this function.

Definition at line 182 of file arch.h.

3.2.2.24 int(∗ yasm_arch_module::set_var)(yasm_arch ∗arch, const char ∗var, unsigned long val)

Module-level implementation of yasm_arch_set_var().

Call yasm_arch_set_var() instead of calling this function.

Definition at line 127 of file arch.h.

3.2.2.25 unsigned int yasm_arch_module::wordsize

Canonical "word" size in bits.

Call yasm_arch_wordsize() to get the word size of a particular yasm_arch.

Definition at line 221 of file arch.h.

The documentation for this struct was generated from the following file:

- libyasm/arch.h

3.3 yasm_assoc_data_callback Struct Reference

YASM associated data callback structure.

`#include <coretype.h>`

Data Fields

- void(∗ destroy)(void ∗data)

  Free memory allocated for associated data.
- void(∗ print)(void ∗data, FILE ∗f, int indent_level)

  Print a description of allocated data.

3.3.1 Detailed Description

YASM associated data callback structure.

Many data structures can have arbitrary data associated with them.

Definition at line 73 of file coretype.h.

3.3.2 Field Documentation

3.3.2.1 void(∗ yasm_assoc_data_callback::destroy)(void ∗data)

Free memory allocated for associated data.

Parameters

data	associated data

Definition at line 77 of file coretype.h.

3.3.2.2 void(∗ yasm_assoc_data_callback::print)(void ∗data, FILE ∗f, int indent_level)

Print a description of allocated data.

For debugging purposes.

Parameters

data	associated data
f	output file
indent_level	indentation level

Definition at line 84 of file coretype.h.

The documentation for this struct was generated from the following file:

- libyasm/coretype.h

3.4 yasm_bytecode Struct Reference

A bytecode.

`#include <bytecode.h>`

Public Member Functions

- STAILQ_ENTRY (yasm_bytecode) link

  Bytecodes are stored as a singly linked list, with tail insertion.

Data Fields

- const yasm_bytecode_callback ∗ callback

  The bytecode callback structure for this bytecode.
- yasm_section ∗ section

Pointer to section containing bytecode; NULL if not part of a section.

- yasm_expr * multiple

  Number of times bytecode is repeated.

- unsigned long len

  Total length of entire bytecode (not including multiple copies).

- long mult_int

  Number of copies, integer version.

- unsigned long line

  Line number where bytecode was defined.

- unsigned long offset

  Offset of bytecode from beginning of its section.

- unsigned long bc_index

  Unique integer index of bytecode.

- yasm_symrec ** symrecs

  NULL-terminated array of labels that point to this bytecode (as the bytecode previous to the label).

- void * contents

  Implementation-specific data (type identified by callback).

3.4.1 Detailed Description

A bytecode.

Definition at line 183 of file bytecode.h.

3.4.2 Member Function Documentation

3.4.2.1 yasm_bytecode::STAILQ_ENTRY (yasm_bytecode)

Bytecodes are stored as a singly linked list, with tail insertion.

See Also

section.h (yasm_section).

3.4.3 Field Documentation

3.4.3.1 unsigned long yasm_bytecode::bc_index

Unique integer index of bytecode.

Used during optimization.

Definition at line 219 of file bytecode.h.

3.4.3.2 const yasm_bytecode_callback* yasm_bytecode::callback

The bytecode callback structure for this bytecode.

May be NULL during partial initialization.

Definition at line 192 of file bytecode.h.

3.4.3.3 void* yasm_bytecode::contents

Implementation-specific data (type identified by callback).

Definition at line 227 of file bytecode.h.

3.4.3.4 unsigned long yasm_bytecode::len

Total length of entire bytecode (not including multiple copies).

Definition at line 205 of file bytecode.h.

3.4.3.5 unsigned long yasm_bytecode::line

Line number where bytecode was defined.

Definition at line 211 of file bytecode.h.

3.4.3.6 long yasm_bytecode::mult_int

Number of copies, integer version.

Definition at line 208 of file bytecode.h.

3.4.3.7 yasm_expr* yasm_bytecode::multiple

Number of times bytecode is repeated.

NULL=1 (to save space in the common case).

Definition at line 202 of file bytecode.h.

3.4.3.8 unsigned long yasm_bytecode::offset

Offset of bytecode from beginning of its section.

0-based, ~0UL (e.g. all 1 bits) if unknown.

Definition at line 216 of file bytecode.h.

3.4.3.9 yasm_symrec** yasm_bytecode::symrecs

NULL-terminated array of labels that point to this bytecode (as the bytecode previous to the label).

NULL if no labels point here.

Definition at line 224 of file bytecode.h.

The documentation for this struct was generated from the following file:

- libyasm/bytecode.h

3.5 yasm_bytecode_callback Struct Reference

Bytecode callback structure.

`#include <bytecode.h>`

Public Types

- enum yasm_bytecode_special_type { **YASM_BC_SPECIAL_NONE** = 0, YASM_BC_SPECIAL_RESERVE, YASM_BC_SPECIAL_OFFSET, YASM_BC_SPECIAL_INSN }

  Special bytecode classifications.

Data Fields

- void(∗ destroy)(void ∗contents)

  Destroys the implementation-specific data.
- void(∗ print)(const void ∗contents, FILE ∗f, int indent_level)

  Prints the implementation-specific data (for debugging purposes).
- void(∗ finalize)(yasm_bytecode ∗bc, yasm_bytecode ∗prev_bc)

  Finalizes the bytecode after parsing.
- int(∗ elem_size)(yasm_bytecode ∗bc)

  Return elements size of a data bytecode.
- int(∗ calc_len)(yasm_bytecode ∗bc, yasm_bc_add_span_func add_span, void ∗add_span_data)

  Calculates the minimum size of a bytecode.
- int(∗ expand)(yasm_bytecode ∗bc, int span, long old_val, long new_val, long ∗neg_thres, long ∗pos_thres)

  Recalculates the bytecode's length based on an expanded span length.
- int(∗ tobytes)(yasm_bytecode ∗bc, unsigned char ∗∗bufp, unsigned char ∗bufstart, void ∗d, yasm_output_value_func output_value, yasm_output_reloc_func output_reloc)

  Convert a bytecode into its byte representation.
- enum yasm_bytecode_callback::yasm_bytecode_special_type **special**

3.5.1 Detailed Description

Bytecode callback structure.

Any implementation of a specific bytecode must implement these functions and this callback structure. The bytecode implementation-specific data is stored in yasm_bytecode.contents.

Definition at line 63 of file bytecode.h.

3.5.2 Member Enumeration Documentation

3.5.2.1 enum yasm_bytecode_callback::yasm_bytecode_special_type

Special bytecode classifications.

Most bytecode types should use #YASM_BC_SPECIAL_NONE. Others cause special handling to kick in in various parts of yasm.

Enumerator

YASM_BC_SPECIAL_RESERVE Bytecode reserves space instead of outputting data.

YASM_BC_SPECIAL_OFFSET Adjusts offset instead of calculating len.

YASM_BC_SPECIAL_INSN Instruction bytecode.

Definition at line 168 of file bytecode.h.

3.5.3 Field Documentation

3.5.3.1 int(* yasm_bytecode_callback::calc_len)(yasm_bytecode *bc, yasm_bc_add_span_func add_span, void *add_span_data)

Calculates the minimum size of a bytecode.

Called from yasm_bc_calc_len(). A generic fill-in for this is yasm_bc_calc_len_common(), but as this function internal errors when called, be very careful when using it! This function should simply add to bc->len and not set it directly (it's initialized by yasm_bc_calc_len() prior to passing control to this function).

Parameters

bc	bytecode
add_span	function to call to add a span
add_span_data	extra data to be passed to add_span function

Returns

0 if no error occurred, nonzero if there was an error recognized (and output) during execution.

Note

May store to bytecode updated expressions.

Definition at line 108 of file bytecode.h.

3.5.3.2 void(* yasm_bytecode_callback::destroy)(void *contents)

Destroys the implementation-specific data.

Called from yasm_bc_destroy().

Parameters

contents	yasm_bytecode.contents

Definition at line 68 of file bytecode.h.

3.5.3.3 int(* yasm_bytecode_callback::elem_size)(yasm_bytecode *bc)

Return elements size of a data bytecode.

This function should return the size of each elements of a data bytecode, for proper dereference of symbols attached to it.

Parameters

bc	bytecode

Returns

0 if element size is unknown.

Definition at line 91 of file bytecode.h.

3.5.3.4 int(* yasm_bytecode_callback::expand)(yasm_bytecode *bc, int span, long old_val, long new_val,long *neg_thres,long *pos_thres)

Recalculates the bytecode's length based on an expanded span length.

Called from yasm_bc_expand(). A generic fill-in for this is yasm_bc_expand_common(), but as this function internal errors when called, if used, ensure that calc_len() never adds a span. This function should simply add to bc->len to increase the length by a delta amount.

Parameters

bc	bytecode
span	span ID (as given to add_span in calc_len)
old_val	previous span value
new_val	new span value
neg_thres	negative threshold for long/short decision (returned)
pos_thres	positive threshold for long/short decision (returned)

Returns

0 if bc no longer dependent on this span's length, negative if there was an error recognized (and output) during execution, and positive if bc size may increase for this span further based on the new negative and positive thresholds returned.

Note

May store to bytecode updated expressions.

Definition at line 132 of file bytecode.h.

3.5.3.5 void(* yasm_bytecode_callback::finalize)(yasm_bytecode *bc, yasm_bytecode *prev_bc)

Finalizes the bytecode after parsing.

Called from yasm_bc_finalize(). A generic fill-in for this is yasm_bc_finalize_common().

Parameters

bc	bytecode
prev_bc	bytecode directly preceding bc

Definition at line 83 of file bytecode.h.

3.5.3.6 void(* yasm_bytecode_callback::print)(const void *contents, FILE *f, int indent_level)

Prints the implementation-specific data (for debugging purposes).

Called from yasm_bc_print().

Parameters

contents	yasm_bytecode.contents
f	file
indent_level	indentation level

Definition at line 76 of file bytecode.h.

3.5.3.7 int(∗ yasm_bytecode_callback::tobytes)(yasm_bytecode ∗bc, unsigned char ∗∗bufp, unsigned char ∗bufstart, void ∗d, yasm_output_value_func output_value,yasm_output_reloc_func output_reloc)

Convert a bytecode into its byte representation.

Called from yasm_bc_tobytes(). A generic fill-in for this is yasm_bc_tobytes_common(), but as this function internal errors when called, be very careful when using it!

Parameters

bc	bytecode
bufp	byte representation destination buffer; should be incremented as it's written to, so that on return its delta from the passed-in buf matches the bytecode length (it's okay not to do this if an error indication is returned)
bufstart	For calculating the correct offset parameter for the *output_value* calls: ∗bufp - bufstart.
d	data to pass to each call to output_value/output_reloc
output_value	function to call to convert values into their byte representation
output_reloc	function to call to output relocation entries for a single sym

Returns

Nonzero on error, 0 on success.

Note

May result in non-reversible changes to the bytecode, but it's preferable if calling this function twice would result in the same output.

Definition at line 159 of file bytecode.h.

The documentation for this struct was generated from the following file:

- libyasm/bytecode.h

3.6 yasm_dbgfmt_module Struct Reference

Debug format module interface.

`#include <dbgfmt.h>`

Data Fields

- const char ∗ name
  One-line description of the debug format.
- const char ∗ keyword
  Keyword used to select debug format.
- const yasm_directive ∗ directives
  NULL-terminated list of directives.
- yasm_dbgfmt ∗(∗ create)(yasm_object ∗object)
  Create debug format.
- void(∗ destroy)(yasm_dbgfmt ∗dbgfmt)
  Module-level implementation of yasm_dbgfmt_destroy().
- void(∗ generate)(yasm_object ∗object, yasm_linemap ∗linemap, yasm_errwarns ∗errwarns)
  Module-level implementation of yasm_dbgfmt_generate().

3.6.1 Detailed Description

Debug format module interface.

Definition at line 44 of file dbgfmt.h.

3.6.2 Field Documentation

3.6.2.1 yasm_dbgfmt*(* yasm_dbgfmt_module::create)(yasm_object *object)

Create debug format.

Module-level implementation of yasm_dbgfmt_create(). The filenames are provided solely for informational purposes.

Parameters

object	object

Returns

NULL if object format does not provide needed support.

Definition at line 60 of file dbgfmt.h.

3.6.2.2 void(* yasm_dbgfmt_module::destroy)(yasm_dbgfmt *dbgfmt)

Module-level implementation of yasm_dbgfmt_destroy().

Call yasm_dbgfmt_destroy() instead of calling this function.

Definition at line 65 of file dbgfmt.h.

3.6.2.3 const yasm_directive* yasm_dbgfmt_module::directives

NULL-terminated list of directives.

NULL if none.

Definition at line 52 of file dbgfmt.h.

3.6.2.4 void(* yasm_dbgfmt_module::generate)(yasm_object *object, yasm_linemap *linemap, yasm_errwarns *errwarns)

Module-level implementation of yasm_dbgfmt_generate().

Call yasm_dbgfmt_generate() instead of calling this function.

Definition at line 70 of file dbgfmt.h.

3.6.2.5 const char* yasm_dbgfmt_module::keyword

Keyword used to select debug format.

Definition at line 49 of file dbgfmt.h.

3.6.2.6 const char* yasm_dbgfmt_module::name

One-line description of the debug format.

Definition at line 46 of file dbgfmt.h.

The documentation for this struct was generated from the following file:

- libyasm/dbgfmt.h

3.7 yasm_dir_help Struct Reference

Directive valparam parse helper structure.

`#include <valparam.h>`

Data Fields

- const char * name

  Value portion of val=param (if needsparam=1), or standalone identifier (if needsparam=0).
- int needsparam

  1 if value requires parameter, 0 if it must not have a parameter.
- int(* helper)(void *obj, yasm_valparam *vp, unsigned long line, void *data, uintptr_t arg)

  Helper callback function if name and parameter existence match.
- size_t off

  Offset added to data pointer passed into yasm_dir_helper() before data pointer is given to yasm_dir_help.helper().
- uintptr_t arg

  Argument to pass in as the arg parameter to yasm_dir_help.helper().

3.7.1 Detailed Description

Directive valparam parse helper structure.

Definition at line 242 of file valparam.h.

3.7.2 Field Documentation

3.7.2.1 int(* yasm_dir_help::helper)(void *obj, yasm_valparam *vp, unsigned long line, void *data, uintptr_t arg)

Helper callback function if name and parameter existence match.

Parameters

obj	obj passed into yasm_dir_helper()
vp	value/parameter
line	line passed into yasm_dir_helper()
data	data passed into yasm_dir_helper() plus yasm_dir_help.off offset
arg	yasm_dir_help.arg argument

Returns

-1 on error, 0 otherwise.

Definition at line 260 of file valparam.h.

3.7.2.2 int yasm_dir_help::needsparam

1 if value requires parameter, 0 if it must not have a parameter.

Definition at line 249 of file valparam.h.

3.7.2.3 size_t yasm_dir_help::off

Offset added to data pointer passed into yasm_dir_helper() before data pointer is given to yasm_dir_help.helper().

This is so that a structure can be passed into yasm_dir_helper() and this can be an offsetof() to point the helper function to a specific structure member.

Definition at line 269 of file valparam.h.

The documentation for this struct was generated from the following file:

- libyasm/valparam.h

3.8 yasm_directive Struct Reference

Directive list entry structure.

`#include <valparam.h>`

Public Types

- enum yasm_directive_flags { YASM_DIR_ANY = 0, YASM_DIR_ARG_REQUIRED = 1, YASM_DIR_ID_REQUIRED = 2 }

  Flags for pre-handler parameter checking.

Data Fields

- const char * name

  Directive name.
- const char * parser

  Parser keyword.
- void(* handler)(yasm_object *object, yasm_valparamhead *valparams, yasm_valparamhead *objext_valparams, unsigned long line)

  Handler callback function for the directive.
- enum yasm_directive::yasm_directive_flags **flags**

3.8.1 Detailed Description

Directive list entry structure.

Definition at line 69 of file valparam.h.

3.8.2 Member Enumeration Documentation

3.8.2.1 enum yasm_directive::yasm_directive_flags

Flags for pre-handler parameter checking.

Enumerator

YASM_DIR_ANY Any valparams accepted.

YASM_DIR_ARG_REQUIRED Require at least 1 valparam.

YASM_DIR_ID_REQUIRED First valparam must be ID.

Definition at line 88 of file valparam.h.

3.8.3 Field Documentation

3.8.3.1 void(∗ yasm_directive::handler)(yasm_object ∗object, yasm_valparamhead ∗valparams, yasm_valparamhead ∗objext_valparams, unsigned long line)

Handler callback function for the directive.

Parameters

object	object
valparams	value/parameters
objext_valparams	object format-specific value/parameters
line	virtual line (from yasm_linemap)

Definition at line 84 of file valparam.h.

3.8.3.2 const char∗ yasm_directive::name

Directive name.

GAS directives should include the ".", NASM directives should just be the raw name (not including the []). NULL entry required to terminate list of directives.

Definition at line 74 of file valparam.h.

The documentation for this struct was generated from the following file:

- libyasm/valparam.h

3.9 yasm_expr Struct Reference

Expression.

`#include <expr.h>`

Data Fields

- yasm_expr_op op

  Operation.
- unsigned long line

  Line number where expression was defined.
- int numterms

  Number of terms in the expression.
- yasm_expr__item terms [2]

  Terms of the expression.

3.9.1 Detailed Description

Expression.

Definition at line 70 of file expr.h.

3.9.2 Field Documentation

3.9.2.1 unsigned long yasm_expr::line

Line number where expression was defined.

Definition at line 72 of file expr.h.

3.9.2.2 int yasm_expr::numterms

Number of terms in the expression.

Definition at line 73 of file expr.h.

3.9.2.3 yasm_expr_op yasm_expr::op

Operation.

Definition at line 71 of file expr.h.

3.9.2.4 yasm_expr__item yasm_expr::terms[2]

Terms of the expression.

Structure may be extended to include more terms, as some operations may allow more than two operand terms (ADD, MUL, OR, AND, XOR).

Definition at line 79 of file expr.h.

The documentation for this struct was generated from the following file:

- libyasm/expr.h

3.10 yasm_expr__item Struct Reference

Expression item.

`#include <expr.h>`

Data Fields

- yasm_expr__type type
  Type.
- union {
  yasm_bytecode * precbc
  Direct bytecode ref (YASM_EXPR_PRECBC)
  yasm_symrec * sym
  Symbol (YASM_EXPR_SYM)
  yasm_expr * expn
  Subexpression (YASM_EXPR_EXPR)

yasm_intnum * intn
Integer value (YASM_EXPR_INT)
yasm_floatnum * flt
Floating point value (YASM_EXPR_FLOAT)
uintptr_t reg
Register (YASM_EXPR_REG)
unsigned int subst
Subst placeholder (YASM_EXPR_SUBST)
} **data**

3.10.1 Detailed Description

Expression item.

Definition at line 54 of file expr.h.

The documentation for this struct was generated from the following file:

- libyasm/expr.h

3.11 yasm_listfmt_module Struct Reference

YASM list format module interface.

`#include <listfmt.h>`

Data Fields

- const char * name
  One-line description of the list format.
- const char * keyword
  Keyword used to select list format.
- yasm_listfmt *(* create)(const char *in_filename, const char *obj_filename)
  Create list format.
- void(* destroy)(yasm_listfmt *listfmt)
  Module-level implementation of yasm_listfmt_destroy().
- void(* output)(yasm_listfmt *listfmt, FILE *f, yasm_linemap *linemap, yasm_arch *arch)
  Module-level implementation of yasm_listfmt_output().

3.11.1 Detailed Description

YASM list format module interface.

Definition at line 44 of file listfmt.h.

3.11.2 Field Documentation

3.11.2.1 yasm_listfmt*(* yasm_listfmt_module::create)(const char *in_filename, const char *obj_filename)

Create list format.

Module-level implementation of yasm_listfmt_create(). The filenames are provided solely for informational purposes.

Parameters

in_filename	primary input filename
obj_filename	object filename

Returns

NULL if unable to initialize.

Definition at line 59 of file listfmt.h.

3.11.2.2 void(* yasm_listfmt_module::destroy)(yasm_listfmt *listfmt)

Module-level implementation of yasm_listfmt_destroy().

Call yasm_listfmt_destroy() instead of calling this function.

Definition at line 64 of file listfmt.h.

3.11.2.3 const char* yasm_listfmt_module::keyword

Keyword used to select list format.

Definition at line 49 of file listfmt.h.

3.11.2.4 const char* yasm_listfmt_module::name

One-line description of the list format.

Definition at line 46 of file listfmt.h.

3.11.2.5 void(* yasm_listfmt_module::output)(yasm_listfmt *listfmt, FILE *f, yasm_linemap *linemap, yasm_arch *arch)

Module-level implementation of yasm_listfmt_output().

Call yasm_listfmt_output() instead of calling this function.

Definition at line 69 of file listfmt.h.

The documentation for this struct was generated from the following file:

- libyasm/listfmt.h

3.12 yasm_object Struct Reference

An object.

`#include <section.h>`

Public Member Functions

- STAILQ_HEAD (yasm_sectionhead, yasm_section) sections

  Linked list of sections.

Data Fields

- char * src_filename
  Source filename.
- char * obj_filename
  Object filename.
- yasm_symtab * symtab
  Symbol table.
- yasm_arch * arch
  Target architecture.
- yasm_objfmt * objfmt
  Object format.
- yasm_dbgfmt * dbgfmt
  Debug format.
- yasm_overrides * overrides
  Function overrides.
- yasm_section * cur_section
  Currently active section.
- struct HAMT * directives
  Directives, organized as two level HAMT; first level is parser, second level is directive name.
- char * global_prefix
  Prefix prepended to externally-visible symbols (empty string if none)
- char * global_suffix
  Suffix appended to externally-visible symbols (empty string if none)

3.12.1 Detailed Description

An object.

This is the internal representation of an object file.

Definition at line 58 of file section.h.

3.12.2 Member Function Documentation

3.12.2.1 yasm_object::STAILQ_HEAD (yasm_sectionhead , yasm_section)

Linked list of sections.

3.12.3 Field Documentation

3.12.3.1 yasm_section* yasm_object::cur_section

Currently active section.

Used by some directives. NULL if no section active.

Definition at line 71 of file section.h.

The documentation for this struct was generated from the following file:

- libyasm/section.h

3.13 yasm_objfmt_module Struct Reference

Object format module interface.

`#include <objfmt.h>`

Data Fields

- const char * name

  One-line description of the object format.
- const char * keyword

  Keyword used to select object format.
- const char * extension

  Default output file extension (without the '.
- const unsigned char default_x86_mode_bits

  Default (starting) x86 BITS setting.
- const unsigned char id_at_ok

  If @ signs should be legal in identifiers.
- const char ** dbgfmt_keywords

  NULL-terminated list of debug format (yasm_dbgfmt) keywords that are valid to use with this object format.
- const char * default_dbgfmt_keyword

  Default debug format keyword (set even if there's only one available to use).
- const yasm_directive * directives

  NULL-terminated list of directives.
- const yasm_stdmac * stdmacs

  NULL-terminated list of standard macro lookups.
- yasm_objfmt *(* create)(yasm_object *object)

  Create object format.
- void(* output)(yasm_object *o, FILE *f, int all_syms, yasm_errwarns *errwarns)

  Module-level implementation of yasm_objfmt_output().
- void(* destroy)(yasm_objfmt *objfmt)

  Module-level implementation of yasm_objfmt_destroy().
- yasm_section *(* add_default_section)(yasm_object *object)

  Module-level implementation of yasm_objfmt_add_default_section().
- void(* init_new_section)(yasm_section *section, unsigned long line)

  Module-level implementation of yasm_objfmt_init_new_section().
- yasm_section *(* section_switch)(yasm_object *object, yasm_valparamhead *valparams, yasm_valparamhead *objext_valparams, unsigned long line)

  Module-level implementation of yasm_objfmt_section_switch().
- yasm_symrec *(* get_special_sym)(yasm_object *object, const char *name, const char *parser)

  Module-level implementation of yasm_objfmt_get_special_sym().

3.13.1 Detailed Description

Object format module interface.

Definition at line 44 of file objfmt.h.

3.13.2 Field Documentation

3.13.2.1 yasm_section*(* yasm_objfmt_module::add_default_section)(yasm_object *object)

Module-level implementation of yasm_objfmt_add_default_section().

Call yasm_objfmt_add_default_section() instead of calling this function.

Definition at line 105 of file objfmt.h.

3.13.2.2 yasm_objfmt*(* yasm_objfmt_module::create)(yasm_object *object)

Create object format.

Module-level implementation of yasm_objfmt_create(). Call yasm_objfmt_create() instead of calling this function.

Parameters

object	object
a	architecture in use

Returns

NULL if architecture/machine combination not supported.

Definition at line 89 of file objfmt.h.

3.13.2.3 const char** yasm_objfmt_module::dbgfmt_keywords

NULL-terminated list of debug format (yasm_dbgfmt) keywords that are valid to use with this object format.

The null debug format (null_dbgfmt, "null") should always be in this list so it's possible to have no debug output.

Definition at line 69 of file objfmt.h.

3.13.2.4 const unsigned char yasm_objfmt_module::default_x86_mode_bits

Default (starting) x86 BITS setting.

This only appies to the x86 architecture; other architectures ignore this setting.

Definition at line 59 of file objfmt.h.

3.13.2.5 void*(* yasm_objfmt_module::destroy)(yasm_objfmt *objfmt)

Module-level implementation of yasm_objfmt_destroy().

Call yasm_objfmt_destroy() instead of calling this function.

Definition at line 100 of file objfmt.h.

3.13.2.6 const yasm_directive* yasm_objfmt_module::directives

NULL-terminated list of directives.

NULL if none.

Definition at line 77 of file objfmt.h.

3.13.2.7 const char* yasm_objfmt_module::extension

Default output file extension (without the '.

'). NULL means no extension, with no '.', while "" includes the '.'.

Definition at line 54 of file objfmt.h.

3.13.2.8 yasm_symrec*(* yasm_objfmt_module::get_special_sym)(yasm_object *object, const char *name, const char *parser)

Module-level implementation of yasm_objfmt_get_special_sym().

Call yasm_objfmt_get_special_sym() instead of calling this function.

Definition at line 124 of file objfmt.h.

3.13.2.9 const unsigned char yasm_objfmt_module::id_at_ok

If @ signs should be legal in identifiers.

Definition at line 62 of file objfmt.h.

3.13.2.10 void(* yasm_objfmt_module::init_new_section)(yasm_section *section, unsigned long line)

Module-level implementation of yasm_objfmt_init_new_section().

Call yasm_objfmt_init_new_section() instead of calling this function.

Definition at line 110 of file objfmt.h.

3.13.2.11 const char* yasm_objfmt_module::keyword

Keyword used to select object format.

Definition at line 49 of file objfmt.h.

3.13.2.12 const char* yasm_objfmt_module::name

One-line description of the object format.

Definition at line 46 of file objfmt.h.

3.13.2.13 void(* yasm_objfmt_module::output)(yasm_object *o, FILE *f, int all_syms, yasm_errwarns *errwarns)

Module-level implementation of yasm_objfmt_output().

Call yasm_objfmt_output() instead of calling this function.

Definition at line 94 of file objfmt.h.

3.13.2.14 yasm_section*(* yasm_objfmt_module::section_switch)(yasm_object *object, yasm_valparamhead *valparams,yasm_valparamhead *objext_valparams, unsigned long line)

Module-level implementation of yasm_objfmt_section_switch().

Call yasm_objfmt_section_switch() instead of calling this function.

Definition at line 116 of file objfmt.h.

3.13.2.15 const yasm_stdmac* yasm_objfmt_module::stdmacs

NULL-terminated list of standard macro lookups.

NULL if none.

Definition at line 80 of file objfmt.h.

The documentation for this struct was generated from the following file:

- libyasm/objfmt.h

3.14 yasm_overrides Struct Reference

Structure of functions that can be overridden.

`#include <section.h>`

Data Fields

- int(∗ value_finalize)(yasm_value ∗value, yasm_bytecode ∗precbc)
  TODO: documentation.

3.14.1 Detailed Description

Structure of functions that can be overridden.

Definition at line 50 of file section.h.

The documentation for this struct was generated from the following file:

- libyasm/section.h

3.15 yasm_valparam::yasm_param Union Reference

Parameter value.

`#include <valparam.h>`

Data Fields

- char ∗ id
  Identifier.
- char ∗ str
  String.
- yasm_expr ∗ e
  Expression.

3.15.1 Detailed Description

Parameter value.

Definition at line 50 of file valparam.h.

The documentation for this union was generated from the following file:

- libyasm/valparam.h

3.16 yasm_parser_module Struct Reference

YASM parser module interface.

`#include <parser.h>`

Data Fields

- const char * name

  One-line description of the parser.
- const char * keyword

  Keyword used to select parser on the command line.
- const char ** preproc_keywords

  NULL-terminated list of preprocessors that are valid to use with this parser.
- const char * default_preproc_keyword

  Default preprocessor.
- const yasm_stdmac * stdmacs

  NULL-terminated list of standard macro lookups.
- void(* do_parse)(yasm_object *object, yasm_preproc *pp, int save_input, yasm_linemap *linemap, yasm_errwarns *errwarns)

  Parse a source file into an object.

3.16.1 Detailed Description

YASM parser module interface.

The "front end" of the assembler.

Definition at line 34 of file parser.h.

3.16.2 Field Documentation

3.16.2.1 const char* yasm_parser_module::default_preproc_keyword

Default preprocessor.

Definition at line 48 of file parser.h.

3.16.2.2 void(* yasm_parser_module::do_parse)(yasm_object *object, yasm_preproc *pp, int save_input, yasm_linemap *linemap, yasm_errwarns *errwarns)

Parse a source file into an object.

Parameters

object	object to parse into (already created)
pp	preprocessor
save_input	nonzero if the parser should save the original lines of source into the object's linemap (via yasm_linemap_add_data()).
errwarns	error/warning set

Note

Parse errors and warnings are stored into errwarns.

Definition at line 63 of file parser.h.

3.16.2.3 const char** yasm_parser_module::preproc_keywords

NULL-terminated list of preprocessors that are valid to use with this parser.

The raw preprocessor (raw_preproc) should always be in this list so it's always possible to have no preprocessing done.

Definition at line 45 of file parser.h.

3.16.2.4 const yasm_stdmac* yasm_parser_module::stdmacs

NULL-terminated list of standard macro lookups.

NULL if none.

Definition at line 51 of file parser.h.

The documentation for this struct was generated from the following file:

- libyasm/parser.h

3.17 yasm_preproc_module Struct Reference

YASM preprocesor module interface.

`#include <preproc.h>`

Data Fields

- const char * name

  One-line description of the preprocessor.
- const char * keyword

  Keyword used to select preprocessor on the command line.
- yasm_preproc *(* create)(const char *in_filename, yasm_symtab *symtab, yasm_linemap *lm, yasm_errwarns *errwarns)

  Create preprocessor.
- void(* destroy)(yasm_preproc *preproc)

  Module-level implementation of yasm_preproc_destroy().
- char *(* get_line)(yasm_preproc *preproc)

  Module-level implementation of yasm_preproc_get_line().
- size_t(* get_included_file)(yasm_preproc *preproc, char *buf, size_t max_size)

  Module-level implementation of yasm_preproc_get_included_file().
- void(* add_include_file)(yasm_preproc *preproc, const char *filename)

  Module-level implementation of yasm_preproc_add_include_file().
- void(* predefine_macro)(yasm_preproc *preproc, const char *macronameval)

  Module-level implementation of yasm_preproc_predefine_macro().
- void(* undefine_macro)(yasm_preproc *preproc, const char *macroname)

  Module-level implementation of yasm_preproc_undefine_macro().
- void(* define_builtin)(yasm_preproc *preproc, const char *macronameval)

Module-level implementation of yasm_preproc_builtin_define().

- void(∗ add_standard)(yasm_preproc ∗preproc, const char ∗∗macros)

  Module-level implementation of yasm_preproc_add_standard().

3.17.1 Detailed Description

YASM preprocesor module interface.

Definition at line 44 of file preproc.h.

3.17.2 Field Documentation

3.17.2.1 void(∗ yasm_preproc_module::add_include_file)(yasm_preproc ∗preproc, const char ∗filename)

Module-level implementation of yasm_preproc_add_include_file().

Call yasm_preproc_add_include_file() instead of calling this function.

Definition at line 88 of file preproc.h.

3.17.2.2 void(∗ yasm_preproc_module::add_standard)(yasm_preproc ∗preproc, const char ∗∗macros)

Module-level implementation of yasm_preproc_add_standard().

Call yasm_preproc_add_standard() instead of calling this function.

Definition at line 108 of file preproc.h.

3.17.2.3 yasm_preproc∗(∗ yasm_preproc_module::create)(const char ∗in_filename, yasm_symtab ∗symtab, yasm_linemap ∗lm, yasm_errwarns ∗errwarns)

Create preprocessor.

Module-level implementation of yasm_preproc_create(). Call yasm_preproc_create() instead of calling this function.

Parameters

in_filename	initial starting filename, or "-" to read from stdin
symtab	symbol table (may be NULL if none)
lm	line mapping repository
errwarns	error/warnning set.

Returns

New preprocessor.

Note

Any preprocessor errors and warnings are stored into errwarns.

Definition at line 64 of file preproc.h.

3.17.2.4 void(∗ yasm_preproc_module::define_builtin)(yasm_preproc ∗preproc, const char ∗macronameval)

Module-level implementation of yasm_preproc_builtin_define().

Call yasm_preproc_builtin_define() instead of calling this function.

Definition at line 103 of file preproc.h.

3.17.2.5 void(* yasm_preproc_module::destroy)(yasm_preproc *preproc)

Module-level implementation of yasm_preproc_destroy().

Call yasm_preproc_destroy() instead of calling this function.

Definition at line 72 of file preproc.h.

3.17.2.6 size_t(* yasm_preproc_module::get_included_file)(yasm_preproc *preproc,char *buf, size_t max_size)

Module-level implementation of yasm_preproc_get_included_file().

Call yasm_preproc_get_included_file() instead of calling this function.

Definition at line 82 of file preproc.h.

3.17.2.7 char*(* yasm_preproc_module::get_line)(yasm_preproc *preproc)

Module-level implementation of yasm_preproc_get_line().

Call yasm_preproc_get_line() instead of calling this function.

Definition at line 77 of file preproc.h.

3.17.2.8 const char* yasm_preproc_module::keyword

Keyword used to select preprocessor on the command line.

Definition at line 49 of file preproc.h.

3.17.2.9 const char* yasm_preproc_module::name

One-line description of the preprocessor.

Definition at line 46 of file preproc.h.

3.17.2.10 void(* yasm_preproc_module::predefine_macro)(yasm_preproc *preproc, const char *macronameval)

Module-level implementation of yasm_preproc_predefine_macro().

Call yasm_preproc_predefine_macro() instead of calling this function.

Definition at line 93 of file preproc.h.

3.17.2.11 void(* yasm_preproc_module::undefine_macro)(yasm_preproc *preproc, const char *macroname)

Module-level implementation of yasm_preproc_undefine_macro().

Call yasm_preproc_undefine_macro() instead of calling this function.

Definition at line 98 of file preproc.h.

The documentation for this struct was generated from the following file:

- libyasm/preproc.h

3.18 yasm_reloc Struct Reference

Public Member Functions

- STAILQ_ENTRY (yasm_reloc) link

  Link to next reloc.

Data Fields

- yasm_intnum * addr
  Offset (address) within section.
- yasm_symrec * sym
  Relocated symbol.

3.18.1 Detailed Description

Definition at line 42 of file section.h.

The documentation for this struct was generated from the following file:

- libyasm/section.h

3.19 yasm_scanner Struct Reference

Re2c scanner state.

```
#include <file.h>
```

Data Fields

- unsigned char * bot
  Bottom of scan buffer.
- unsigned char * tok
  Start of token.
- unsigned char * ptr
  Scan marker.
- unsigned char * cur
  Cursor (1 past end of token)
- unsigned char * lim
  Limit of good data.
- unsigned char * top
  Top of scan buffer.
- unsigned char * eof
  End of file.

3.19.1 Detailed Description

Re2c scanner state.

Definition at line 38 of file file.h.

The documentation for this struct was generated from the following file:

- libyasm/file.h

3.20 yasm_stdmac Struct Reference

Standard macro structure for modules that allows association of a set of standard macros with a parser/preprocessor combination.

`#include <coretype.h>`

Data Fields

- const char * parser
  Parser keyword.
- const char * preproc
  Preprocessor keyword.
- const char ** macros
  NULL-terminated array of standard macros.

3.20.1 Detailed Description

Standard macro structure for modules that allows association of a set of standard macros with a parser/preprocessor combination.

A NULL-terminated array of these structures is used in a number of module interfaces.

Definition at line 60 of file coretype.h.

3.20.2 Field Documentation

3.20.2.1 const char** yasm_stdmac::macros

NULL-terminated array of standard macros.

May be NULL if no standard macros should be added for this preprocessor.

Definition at line 67 of file coretype.h.

The documentation for this struct was generated from the following file:

- libyasm/coretype.h

3.21 yasm_valparam Struct Reference

Value/parameter pair.

`#include <valparam.h>`

Data Structures

- union yasm_param
  Parameter value.

Public Types

- enum yasm_param_type { YASM_PARAM_ID, YASM_PARAM_STRING, YASM_PARAM_EXPR }

  Parameter type.

Public Member Functions

- STAILQ_ENTRY (yasm_valparam) link

  Next pair in list.

Data Fields

- char * val

  Value.
- enum yasm_valparam::yasm_param_type type

  Parameter type.
- union yasm_valparam::yasm_param param

  Parameter.
- char id_prefix

  Prefix character that indicates a raw identifier.

3.21.1 Detailed Description

Value/parameter pair.

Definition at line 38 of file valparam.h.

3.21.2 Member Enumeration Documentation

3.21.2.1 enum yasm_valparam::yasm_param_type

Parameter type.

Enumerator

YASM_PARAM_ID Identifier.

YASM_PARAM_STRING String.

YASM_PARAM_EXPR Expression.

Definition at line 43 of file valparam.h.

3.21.3 Field Documentation

3.21.3.1 char yasm_valparam::id_prefix

Prefix character that indicates a raw identifier.

When yasm_vp_string() is called on a YASM_PARAM_ID, all characters are returned. When yasm_vp_id() is called on a YASM_PARAM_ID, if the identifier begins with this character, this character is stripped from the returned value.

Definition at line 62 of file valparam.h.

The documentation for this struct was generated from the following file:

- libyasm/valparam.h

3.22 yasm_value Struct Reference

A value.

`#include <coretype.h>`

Data Fields

- yasm_expr * abs
  The absolute portion of the value.
- yasm_symrec * rel
  The relative portion of the value.
- yasm_symrec * wrt
  What the relative portion is in reference to.
- unsigned int seg_of: 1
  If the segment of the relative portion should be used, not the relative portion itself.
- unsigned int rshift: 7
  If the relative portion of the value should be shifted right (supported only by a few object formats).
- unsigned int curpos_rel: 1
  Indicates the relative portion of the value should be relocated relative to the current assembly position rather than relative to the section start.
- unsigned int ip_rel: 1
  Indicates that curpos_rel was set due to IP-relative relocation; in some objfmt/arch combinations (e.g.
- unsigned int jump_target: 1
  Indicates the value is a jump target address (rather than a simple data address).
- unsigned int section_rel: 1
  Indicates the relative portion of the value should be relocated relative to its own section start rather than relative to the section start of the bytecode containing this value.
- unsigned int no_warn: 1
  Indicates overflow warnings have been disabled for this value.
- unsigned int sign: 1
  Sign of the value.
- unsigned int size: 8
  Size of the value, in bits.

3.22.1 Detailed Description

A value.

May be absolute or relative. Outside the parser, yasm_expr should only be used for absolute exprs. Anything that could contain a relocatable value should use this structure instead.

See Also

value.h for related functions.

Definition at line 121 of file coretype.h.

3.22.2 Field Documentation

3.22.2.1 yasm_expr* yasm_value::abs

The absolute portion of the value.

May contain *differences* between symrecs but not standalone symrecs. May be NULL if there is no absolute portion (e.g. the absolute portion is 0).

Definition at line 126 of file coretype.h.

3.22.2.2 unsigned int yasm_value::curpos_rel

Indicates the relative portion of the value should be relocated relative to the current assembly position rather than relative to the section start.

"Current assembly position" here refers to the starting address of the bytecode containing this value. Boolean.

Definition at line 152 of file coretype.h.

3.22.2.3 unsigned int yasm_value::ip_rel

Indicates that curpos_rel was set due to IP-relative relocation; in some objfmt/arch combinations (e.g.

win64/x86-amd64) this info is needed to generate special relocations.

Definition at line 158 of file coretype.h.

3.22.2.4 unsigned int yasm_value::jump_target

Indicates the value is a jump target address (rather than a simple data address).

In some objfmt/arch combinations (e.g. macho/amd64) this info is needed to generate special relocations.

Definition at line 164 of file coretype.h.

3.22.2.5 unsigned int yasm_value::no_warn

Indicates overflow warnings have been disabled for this value.

Definition at line 175 of file coretype.h.

3.22.2.6 yasm_symrec* yasm_value::rel

The relative portion of the value.

This is the portion that may need to generate a relocation. May be NULL if no relative portion.

Definition at line 131 of file coretype.h.

3.22.2.7 unsigned int yasm_value::rshift

If the relative portion of the value should be shifted right (supported only by a few object formats).

If just the absolute portion should be shifted, that must be in the abs expr, not here!

Definition at line 145 of file coretype.h.

3.22.2.8 unsigned int yasm_value::section_rel

Indicates the relative portion of the value should be relocated relative to its own section start rather than relative to the section start of the bytecode containing this value.

E.g. the value resulting from the relative portion should be the offset from its section start. Boolean.

Definition at line 172 of file coretype.h.

3.22.2.9 unsigned int yasm_value::seg_of

If the segment of the relative portion should be used, not the relative portion itself.

Boolean.

Definition at line 139 of file coretype.h.

3.22.2.10 unsigned int yasm_value::sign

Sign of the value.

Nonzero if the final value should be treated as signed, 0 if it should be treated as signed.

Definition at line 180 of file coretype.h.

3.22.2.11 unsigned int yasm_value::size

Size of the value, in bits.

Definition at line 183 of file coretype.h.

3.22.2.12 yasm_symrec* yasm_value::wrt

What the relative portion is in reference to.

NULL if the default.

Definition at line 134 of file coretype.h.

The documentation for this struct was generated from the following file:

- libyasm/coretype.h

4 File Documentation

4.1 libyasm.h File Reference

YASM library primary header file.

```
#include <stdio.h>
#include <stdarg.h>
#include <libyasm-stdint.h>
#include <libyasm/compat-queue.h>
#include <libyasm/coretype.h>
#include <libyasm/valparam.h>
#include <libyasm/linemap.h>
#include <libyasm/errwarn.h>
#include <libyasm/intnum.h>
#include <libyasm/floatnum.h>
#include <libyasm/expr.h>
#include <libyasm/value.h>
#include <libyasm/symrec.h>
#include <libyasm/bytecode.h>
#include <libyasm/section.h>
#include <libyasm/insn.h>
#include <libyasm/arch.h>
#include <libyasm/dbgfmt.h>
#include <libyasm/objfmt.h>
#include <libyasm/listfmt.h>
#include <libyasm/parser.h>
#include <libyasm/preproc.h>
#include <libyasm/file.h>
#include <libyasm/module.h>
#include <libyasm/hamt.h>
#include <libyasm/md5.h>
```

4.1.1 Detailed Description

YASM library primary header file.

Definition in file libyasm.h.

4.2 libyasm/arch.h File Reference

YASM architecture interface.

Data Structures

- struct yasm_arch_machine

  YASM machine subtype.
- struct yasm_arch_module

  YASM architecture module interface.

Typedefs

- typedef enum yasm_arch_create_error yasm_arch_create_error

  Errors that may be returned by yasm_arch_module::create().
- typedef enum yasm_arch_insnprefix yasm_arch_insnprefix

  Return values for yasm_arch_module::parse_check_insnprefix().

- typedef enum yasm_arch_regtmod yasm_arch_regtmod

  Types of registers / target modifiers that may be returned by yasm_arch_module::parse_check_regtmod().
- typedef struct yasm_arch_machine yasm_arch_machine

  YASM machine subtype.
- typedef struct yasm_arch_module yasm_arch_module

  YASM architecture module interface.

Enumerations

- enum yasm_arch_create_error { YASM_ARCH_CREATE_OK = 0, YASM_ARCH_CREATE_BAD_MACHINE, YASM_ARCH_CREATE_BAD_PARSER }

  Errors that may be returned by yasm_arch_module::create().
- enum yasm_arch_insnprefix { YASM_ARCH_NOTINSNPREFIX = 0, YASM_ARCH_INSN, YASM_ARCH_PREFIX }

  Return values for yasm_arch_module::parse_check_insnprefix().
- enum yasm_arch_regtmod {
  YASM_ARCH_NOTREGTMOD = 0, YASM_ARCH_REG, YASM_ARCH_REGGROUP, YASM_ARCH_SEGREG,
  YASM_ARCH_TARGETMOD }

  Types of registers / target modifiers that may be returned by yasm_arch_module::parse_check_regtmod().

Functions

- const char * yasm_arch_name (const yasm_arch *arch)

  Get the one-line description of an architecture.
- const char * yasm_arch_keyword (const yasm_arch *arch)

  Get the keyword used to select an architecture.
- unsigned int yasm_arch_wordsize (const yasm_arch *arch)

  Get the word size of an architecture.
- unsigned int yasm_arch_min_insn_len (const yasm_arch *arch)

  Get the minimum instruction length of an architecture.
- yasm_arch * yasm_arch_create (const yasm_arch_module *module, const char *machine, const char *parser, yasm_arch_create_error *error)

  Create architecture.
- void yasm_arch_destroy (yasm_arch *arch)

  Clean up, free any architecture-allocated memory.
- const char * yasm_arch_get_machine (const yasm_arch *arch)

  Get architecture's active machine name.
- unsigned int yasm_arch_get_address_size (const yasm_arch *arch)

  Get architecture's active address size, in bits.
- int yasm_arch_set_var (yasm_arch *arch, const char *var, unsigned long val)

  Set any arch-specific variables.
- yasm_arch_insnprefix yasm_arch_parse_check_insnprefix (yasm_arch *arch, const char *id, size_t id_len, unsigned long line, yasm_bytecode **bc, uintptr_t *prefix)

  Check an generic identifier to see if it matches architecture specific names for instructions or instruction prefixes.
- yasm_arch_regtmod yasm_arch_parse_check_regtmod (yasm_arch *arch, const char *id, size_t id_len, uintptr_t *data)

  Check an generic identifier to see if it matches architecture specific names for registers or target modifiers.

- const unsigned char ** yasm_arch_get_fill (const yasm_arch *arch)

  Get NOP fill patterns for 1-15 bytes of fill.
- int yasm_arch_floatnum_tobytes (yasm_arch *arch, const yasm_floatnum *flt, unsigned char *buf, size_t destsize, size_t valsize, size_t shift, int warn)

  Output yasm_floatnum to buffer.
- int yasm_arch_intnum_tobytes (yasm_arch *arch, const yasm_intnum *intn, unsigned char *buf, size_t destsize, size_t valsize, int shift, const yasm_bytecode *bc, int warn)

  Output yasm_intnum to buffer.
- unsigned int yasm_arch_get_reg_size (yasm_arch *arch, uintptr_t reg)

  Get the equivalent size of a register in bits.
- uintptr_t yasm_arch_reggroup_get_reg (yasm_arch *arch, uintptr_t reggroup, unsigned long regindex)

  Get a specific register of a register group, based on the register group and the index within the group.
- void yasm_arch_reg_print (yasm_arch *arch, uintptr_t reg, FILE *f)

  Print a register.
- void yasm_arch_segreg_print (yasm_arch *arch, uintptr_t segreg, FILE *f)

  Print a segment register.
- yasm_effaddr * yasm_arch_ea_create (yasm_arch *arch, yasm_expr *e)

  Create an effective address from an expression.
- void yasm_arch_ea_destroy (yasm_arch *arch, yasm_effaddr *ea)

  Delete (free allocated memory for) an effective address.
- void yasm_arch_ea_print (const yasm_arch *arch, const yasm_effaddr *ea, FILE *f, int indent_level)

  Print an effective address.
- yasm_bytecode * yasm_arch_create_empty_insn (yasm_arch *arch, unsigned long line)

  Create a bytecode that represents a single empty (0 length) instruction.

4.2.1 Detailed Description

YASM architecture interface.

Definition in file arch.h.

4.2.2 Typedef Documentation

4.2.2.1 typedef enum yasm_arch_create_error yasm_arch_create_error

Errors that may be returned by yasm_arch_module::create().

4.2.2.2 typedef enum yasm_arch_insnprefix yasm_arch_insnprefix

Return values for yasm_arch_module::parse_check_insnprefix().

4.2.2.3 typedef struct yasm_arch_machine yasm_arch_machine

YASM machine subtype.

A number of different machine types may be associated with a single architecture. These may be specific CPU's, but the ABI used to interface with the architecture should be the primary differentiator between machines. Some object formats (ELF) use the machine to determine parameters within the generated output.

4.2.2.4 typedef struct yasm_arch_module yasm_arch_module

YASM architecture module interface.

Note

All "data" in parser-related functions (yasm_arch_parse_∗) needs to start the parse initialized to 0 to make it okay for a parser-related function to use/check previously stored data to see if it's been called before on the same piece of data.

4.2.3 Enumeration Type Documentation

4.2.3.1 enum yasm_arch_create_error

Errors that may be returned by yasm_arch_module::create().

Enumerator

YASM_ARCH_CREATE_OK No error.

YASM_ARCH_CREATE_BAD_MACHINE Unrecognized machine name.

YASM_ARCH_CREATE_BAD_PARSER Unrecognized parser name.

Definition at line 34 of file arch.h.

4.2.3.2 enum yasm_arch_insnprefix

Return values for yasm_arch_module::parse_check_insnprefix().

Enumerator

YASM_ARCH_NOTINSNPREFIX Unrecognized.

YASM_ARCH_INSN An instruction.

YASM_ARCH_PREFIX An instruction prefix.

Definition at line 41 of file arch.h.

4.2.3.3 enum yasm_arch_regtmod

Types of registers / target modifiers that may be returned by yasm_arch_module::parse_check_regtmod().

Enumerator

YASM_ARCH_NOTREGTMOD Unrecognized.

YASM_ARCH_REG A "normal" register.

YASM_ARCH_REGGROUP A group of indexable registers.

YASM_ARCH_SEGREG A segment register.

YASM_ARCH_TARGETMOD A target modifier (for jumps)

Definition at line 50 of file arch.h.

4.2.4 Function Documentation

4.2.4.1 yasm_arch* yasm_arch_create (const yasm_arch_module * *module*, const char * *machine*, const char * *parser*, yasm_arch_create_error * *error*)

Create architecture.

Parameters

module	architecture module
machine	keyword of machine in use (must be one listed in yasm_arch_module.machines)
parser	keyword of parser in use
error	error return value

Returns

NULL on error (error returned in error parameter), otherwise new architecture.

4.2.4.2 yasm_bytecode* yasm_arch_create_empty_insn (yasm_arch * *arch*, unsigned long *line*)

Create a bytecode that represents a single empty (0 length) instruction.

This is used for handling solitary prefixes.

Parameters

arch	architecture
line	virtual line (from yasm_linemap)

Returns

Newly allocated bytecode.

4.2.4.3 void yasm_arch_destroy (yasm_arch * *arch*)

Clean up, free any architecture-allocated memory.

Parameters

arch	architecture

4.2.4.4 yasm_effaddr* yasm_arch_ea_create (yasm_arch * *arch*, yasm_expr * *e*)

Create an effective address from an expression.

Parameters

arch	architecture
e	expression (kept, do not delete)

Returns

Newly allocated effective address.

4.2.4.5 void yasm_arch_ea_destroy (yasm_arch * *arch*, yasm_effaddr * *ea*)

Delete (free allocated memory for) an effective address.

Parameters

arch	architecture
ea	effective address (only pointer to it).

4.2.4.6 void yasm_arch_ea_print (const yasm_arch * *arch*, const yasm_effaddr * *ea*, FILE * *f*, int *indent_level*)

Print an effective address.

For debugging purposes.

Parameters

arch	architecture
ea	effective address
f	file
indent_level	indentation level

4.2.4.7 int yasm_arch_floatnum_tobytes (yasm_arch * *arch*, const yasm_floatnum * *flt*, unsigned char * *buf*, size_t *destsize*, size_t *valsize*, size_t *shift*, int *warn*)

Output yasm_floatnum to buffer.

Puts the value into the least significant bits of the destination, or may be shifted into more significant bits by the shift parameter. The destination bits are cleared before being set. Architecture-specific because of endianness.

Parameters

arch	architecture
flt	floating point value
buf	buffer to write into
destsize	destination size (in bytes)
valsize	size (in bits)
shift	left shift (in bits)
warn	enables standard overflow/underflow warnings

Returns

Nonzero on error.

4.2.4.8 unsigned int yasm_arch_get_address_size (const yasm_arch * *arch*)

Get architecture's active address size, in bits.

Parameters

arch	architecture

Returns

Active address size (in bits).

4.2.4.9 const unsigned char** yasm_arch_get_fill (const yasm_arch * *arch*)

Get NOP fill patterns for 1-15 bytes of fill.

Parameters

arch	architecture

Returns

16-entry array of arrays; [0] is unused, [1] - [15] point to arrays of 1-15 bytes (respectively) in length.

4.2.4.10 const char* yasm_arch_get_machine (const yasm_arch * *arch*)

Get architecture's active machine name.

Parameters

arch	architecture

Returns

Active machine name.

4.2.4.11 unsigned int yasm_arch_get_reg_size (yasm_arch * *arch,* uintptr_t *reg*)

Get the equivalent size of a register in bits.

Parameters

arch	architecture
reg	register

Returns

0 if there is no suitable equivalent size, otherwise the size.

4.2.4.12 int yasm_arch_intnum_tobytes (yasm_arch * *arch,* const yasm_intnum * *intn,* unsigned char * *buf,* size_t *destsize,* size_t *valsize,* int *shift,* const yasm_bytecode * *bc,* int *warn*)

Output yasm_intnum to buffer.

Puts the value into the least significant bits of the destination, or may be shifted into more significant bits by the shift parameter. The destination bits are cleared before being set.

Parameters

arch	architecture
intn	integer value
buf	buffer to write into
destsize	destination size (in bytes)

valsize	size (in bits)
shift	left shift (in bits); may be negative to specify right shift (standard warnings include truncation to boundary)
bc	bytecode being output ("parent" of value)
warn	enables standard warnings (value doesn't fit into valsize bits)

Returns

Nonzero on error.

4.2.4.13 const char* yasm_arch_keyword (const yasm_arch * *arch*)

Get the keyword used to select an architecture.

Parameters

arch	architecture

Returns

Architecture keyword.

4.2.4.14 unsigned int yasm_arch_min_insn_len (const yasm_arch * *arch*)

Get the minimum instruction length of an architecture.

Parameters

arch	architecture

Returns

Minimum instruction length (in bytes).

4.2.4.15 const char* yasm_arch_name (const yasm_arch * *arch*)

Get the one-line description of an architecture.

Parameters

arch	architecture

Returns

One-line description of architecture.

4.2.4.16 yasm_arch_insnprefix yasm_arch_parse_check_insnprefix (yasm_arch * *arch*, const char * *id*, size_t *id_len*, unsigned long *line*, yasm_bytecode ** *bc*, uintptr_t * *prefix*)

Check an generic identifier to see if it matches architecture specific names for instructions or instruction prefixes.

Unrecognized identifiers should return YASM_ARCH_NOTINSNPREFIX so they can be treated as normal symbols.

Any additional data beyond just the type (almost always necessary) should be returned into the space provided by the data parameter.

Parameters

arch	architecture
id	identifier as in the input file
id_len	length of id string
line	virtual line
bc	for instructions, yasm_insn-based bytecode is returned (and NULL otherwise)
prefix	for prefixes, yasm_arch-specific value is returned (and 0 otherwise)

Returns

Identifier type (YASM_ARCH_NOTINSNPREFIX if unrecognized)

4.2.4.17 yasm_arch_regtmod yasm_arch_parse_check_regtmod (yasm_arch * *arch*, const char * *id*, size_t *id_len*, uintptr_t * *data*)

Check an generic identifier to see if it matches architecture specific names for registers or target modifiers.

Unrecognized identifiers should return YASM_ARCH_NOTREGTMOD. Any additional data beyond just the type (almost always necessary) should be returned into the space provided by the data parameter.

Parameters

arch	architecture
id	identifier as in the input file
id_len	length of id string
data	extra identification information (yasm_arch-specific) [output]

Returns

Identifier type (YASM_ARCH_NOTREGTMOD if unrecognized)

4.2.4.18 void yasm_arch_reg_print (yasm_arch * *arch*, uintptr_t *reg*, FILE * *f*)

Print a register.

For debugging purposes.

Parameters

arch	architecture
reg	register
f	file

4.2.4.19 uintptr_t yasm_arch_reggroup_get_reg (yasm_arch * *arch*, uintptr_t *reggroup*, unsigned long *regindex*)

Get a specific register of a register group, based on the register group and the index within the group.

Parameters

arch	architecture
reggroup	register group

regindex	register index

Returns

0 if regindex is not valid for that register group, otherwise the specific register value.

4.2.4.20 void yasm_arch_segreg_print (yasm_arch * *arch,* uintptr_t *segreg,* FILE * *f*)

Print a segment register.

For debugging purposes.

Parameters

arch	architecture
segreg	segment register
f	file

4.2.4.21 int yasm_arch_set_var (yasm_arch * *arch,* const char * *var,* unsigned long *val*)

Set any arch-specific variables.

For example, "mode_bits" in x86.

Parameters

arch	architecture
var	variable name
val	value to set

Returns

Zero on success, non-zero on failure (variable does not exist).

4.2.4.22 unsigned int yasm_arch_wordsize (const yasm_arch * *arch*)

Get the word size of an architecture.

Parameters

arch	architecture

Returns

Word size (in bits).

4.3 libyasm/assocdat.h File Reference

YASM associated data storage (libyasm internal use)

Typedefs

- typedef struct yasm__assoc_data yasm__assoc_data

  Associated data container.

Functions

- YASM_LIB_DECL yasm__assoc_data * yasm__assoc_data_create (void)

  Create an associated data container.
- YASM_LIB_DECL void * yasm__assoc_data_get (yasm__assoc_data *assoc_data, const yasm_assoc_data_callback *callback)

  Get associated data for a data callback.
- YASM_LIB_DECL yasm__assoc_data * yasm__assoc_data_add (yasm__assoc_data *assoc_data, const yasm_assoc_data_callback *callback, void *data)

  Add associated data to a associated data container.
- YASM_LIB_DECL void yasm__assoc_data_destroy (yasm__assoc_data *assoc_data)

  Destroy all associated data in a container.
- YASM_LIB_DECL void yasm__assoc_data_print (const yasm__assoc_data *assoc_data, FILE *f, int indent_level)

  Print all associated data in a container.

4.3.1 Detailed Description

YASM associated data storage (libyasm internal use)

Definition in file assocdat.h.

4.3.2 Typedef Documentation

4.3.2.1 typedef struct yasm__assoc_data yasm__assoc_data

Associated data container.

Definition at line 38 of file assocdat.h.

4.3.3 Function Documentation

4.3.3.1 YASM_LIB_DECL yasm__assoc_data* yasm__assoc_data_add (yasm__assoc_data * *assoc_data,* const yasm_assoc_data_callback * *callback,* void * *data*)

Add associated data to a associated data container.

Attention

Deletes any existing associated data for that data callback.

Parameters

assoc_data	container of associated data
callback	callback
data	data to associate

4.3.3.2 YASM_LIB_DECL yasm__assoc_data* yasm__assoc_data_create (void)

Create an associated data container.

4.3.3.3 YASM_LIB_DECL void yasm__assoc_data_destroy (yasm__assoc_data * *assoc_data*)

Destroy all associated data in a container.

4.3.3.4 YASM_LIB_DECL void* yasm__assoc_data_get (yasm__assoc_data * *assoc_data,* const yasm_assoc_data_callback * *callback*)

Get associated data for a data callback.

Parameters

assoc_data	container of associated data
callback	callback used when adding data

Returns

Associated data (NULL if none).

4.3.3.5 YASM_LIB_DECL void yasm__assoc_data_print (const yasm__assoc_data * *assoc_data,* FILE * *f,* int *indent_level*)

Print all associated data in a container.

4.4 libyasm/bytecode.h File Reference

YASM bytecode interface.

Data Structures

- struct yasm_bytecode_callback
  Bytecode callback structure.
- struct yasm_bytecode
  A bytecode.

Macros

- #define **YASM_LIB_DECL**
- #define yasm_bc__next(bc) STAILQ_NEXT(bc, link)
  Get the next bytecode in a linked list of bytecodes.

Typedefs

- typedef struct yasm_dataval yasm_dataval
  A data value (opaque type).
- typedef struct yasm_datavalhead yasm_datavalhead
  A list of data values.
- typedef void(* yasm_bc_add_span_func)(void *add_span_data, yasm_bytecode *bc, int id, const yasm_value *value, long neg_thres, long pos_thres)

Add a dependent span for a bytecode.

- typedef struct yasm_bytecode_callback yasm_bytecode_callback

  Bytecode callback structure.

Functions

- STAILQ_HEAD (yasm_datavalhead, yasm_dataval)

  Linked list of data values.
- YASM_LIB_DECL yasm_bytecode * yasm_bc_create_common (const yasm_bytecode_callback *callback, void *contents, unsigned long line)

  Create a bytecode of any specified type.
- YASM_LIB_DECL void yasm_bc_transform (yasm_bytecode *bc, const yasm_bytecode_callback *callback, void *contents)

  Transform a bytecode of any type into a different type.
- YASM_LIB_DECL void yasm_bc_finalize_common (yasm_bytecode *bc, yasm_bytecode *prev_bc)

  Common bytecode callback finalize function, for where no finalization is ever required for this type of bytecode.
- YASM_LIB_DECL int yasm_bc_calc_len_common (yasm_bytecode *bc, yasm_bc_add_span_func add_span, void *add_span_data)

  Common bytecode callback calc_len function, for where the bytecode has no calculatable length.
- YASM_LIB_DECL int yasm_bc_expand_common (yasm_bytecode *bc, int span, long old_val, long new_val, long *neg_thres, long *pos_thres)

  Common bytecode callback expand function, for where the bytecode is always short (calc_len never calls add_span).
- YASM_LIB_DECL int yasm_bc_tobytes_common (yasm_bytecode *bc, unsigned char **bufp, unsigned char *bufstart, void *d, yasm_output_value_func output_value, yasm_output_reloc_func output_reloc)

  Common bytecode callback tobytes function, for where the bytecode cannot be converted to bytes.
- YASM_LIB_DECL void yasm_bc_set_multiple (yasm_bytecode *bc, yasm_expr *e)

  Set multiple field of a bytecode.
- YASM_LIB_DECL yasm_bytecode * yasm_bc_create_data (yasm_datavalhead *datahead, unsigned int size, int append_zero, yasm_arch *arch, unsigned long line)

  Create a bytecode containing data value(s).
- YASM_LIB_DECL yasm_bytecode * yasm_bc_create_leb128 (yasm_datavalhead *datahead, int sign, unsigned long line)

  Create a bytecode containing LEB128-encoded data value(s).
- YASM_LIB_DECL yasm_bytecode * yasm_bc_create_reserve (yasm_expr *numitems, unsigned int itemsize, unsigned long line)

  Create a bytecode reserving space.
- YASM_LIB_DECL const yasm_expr * yasm_bc_reserve_numitems (yasm_bytecode *bc, unsigned int *itemsize)

  Get the number of items and itemsize for a reserve bytecode.
- YASM_LIB_DECL yasm_bytecode * yasm_bc_create_incbin (char *filename, yasm_expr *start, yasm_expr *maxlen, yasm_linemap *linemap, unsigned long line)

  Create a bytecode that includes a binary file verbatim.
- YASM_LIB_DECL yasm_bytecode * yasm_bc_create_align (yasm_expr *boundary, yasm_expr *fill, yasm_expr *maxskip, const unsigned char **code_fill, unsigned long line)

  Create a bytecode that aligns the following bytecode to a boundary.
- YASM_LIB_DECL yasm_bytecode * yasm_bc_create_org (unsigned long start, unsigned long fill, unsigned long line)

  Create a bytecode that puts the following bytecode at a fixed section offset.

- YASM_LIB_DECL yasm_section * yasm_bc_get_section (yasm_bytecode *bc)

  Get the section that contains a particular bytecode.
- YASM_LIB_DECL void yasm_bc__add_symrec (yasm_bytecode *bc, yasm_symrec *sym)

  Add to the list of symrecs that reference a bytecode.
- YASM_LIB_DECL void yasm_bc_destroy (yasm_bytecode *bc)

  Delete (free allocated memory for) a bytecode.
- YASM_LIB_DECL void yasm_bc_print (const yasm_bytecode *bc, FILE *f, int indent_level)

  Print a bytecode.
- YASM_LIB_DECL void yasm_bc_finalize (yasm_bytecode *bc, yasm_bytecode *prev_bc)

  Finalize a bytecode after parsing.
- YASM_LIB_DECL yasm_intnum * yasm_calc_bc_dist (yasm_bytecode *precbc1, yasm_bytecode *precbc2)

  Determine the distance between the starting offsets of two bytecodes.
- YASM_LIB_DECL unsigned long yasm_bc_next_offset (yasm_bytecode *precbc)

  Get the offset of the next bytecode (the next bytecode doesn't have to actually exist).
- YASM_LIB_DECL int yasm_bc_elem_size (yasm_bytecode *bc)

  Return elemens size of a data bytecode.
- YASM_LIB_DECL int yasm_bc_calc_len (yasm_bytecode *bc, yasm_bc_add_span_func add_span, void *add_span_data)

  Resolve EQUs in a bytecode and calculate its minimum size.
- YASM_LIB_DECL int yasm_bc_expand (yasm_bytecode *bc, int span, long old_val, long new_val, long *neg_thres, long *pos_thres)

  Recalculate a bytecode's length based on an expanded span length.
- YASM_LIB_DECL unsigned char * yasm_bc_tobytes (yasm_bytecode *bc, unsigned char *buf, unsigned long *bufsize, int *gap, void *d, yasm_output_value_func output_value, yasm_output_reloc_func output_reloc)

  Convert a bytecode into its byte representation.
- YASM_LIB_DECL int yasm_bc_get_multiple (yasm_bytecode *bc, long *multiple, int calc_bc_dist)

  Get the bytecode multiple value as an integer.
- YASM_LIB_DECL const yasm_expr * yasm_bc_get_multiple_expr (const yasm_bytecode *bc)

  Get the bytecode multiple value as an expression.
- YASM_LIB_DECL yasm_insn * yasm_bc_get_insn (yasm_bytecode *bc)

  Get a yasm_insn structure from an instruction bytecode (if possible).
- YASM_LIB_DECL yasm_dataval * yasm_dv_create_expr (yasm_expr *expn)

  Create a new data value from an expression.
- YASM_LIB_DECL yasm_dataval * yasm_dv_create_string (char *contents, size_t len)

  Create a new data value from a string.
- YASM_LIB_DECL yasm_dataval * yasm_dv_create_raw (unsigned char *contents, unsigned long len)

  Create a new data value from raw bytes data.
- YASM_LIB_DECL yasm_dataval * yasm_dv_create_reserve (void)

  Create a new uninitialized data value.
- YASM_LIB_DECL yasm_value * yasm_dv_get_value (yasm_dataval *dv)

  Get the underlying value of a data value.
- YASM_LIB_DECL void yasm_dv_set_multiple (yasm_dataval *dv, yasm_expr *e)

  Set multiple field of a data value.
- YASM_LIB_DECL int yasm_dv_get_multiple (yasm_dataval *dv, unsigned long *multiple)

  Get the data value multiple value as an unsigned long integer.
- void yasm_dvs_initialize (yasm_datavalhead *headp)

  Initialize a list of data values.

- YASM_LIB_DECL void yasm_dvs_delete (yasm_datavalhead *headp)

  Delete (free allocated memory for) a list of data values.
- YASM_LIB_DECL yasm_dataval * yasm_dvs_append (yasm_datavalhead *headp, yasm_dataval *dv)

  Add data value to the end of a list of data values.
- YASM_LIB_DECL void yasm_dvs_print (const yasm_datavalhead *headp, FILE *f, int indent_level)

  Print a data value list.

4.4.1 Detailed Description

YASM bytecode interface.

Definition in file bytecode.h.

4.4.2 Macro Definition Documentation

4.4.2.1 #define yasm_bc__next(*bc*) STAILQ_NEXT(bc, link)

Get the next bytecode in a linked list of bytecodes.

Parameters

bc	bytecode

Returns

Next bytecode.

Definition at line 287 of file bytecode.h.

4.4.3 Typedef Documentation

4.4.3.1 typedef void(* yasm_bc_add_span_func)(void *add_span_data, yasm_bytecode *bc, int id, const yasm_value *value, long neg_thres, long pos_thres)

Add a dependent span for a bytecode.

Parameters

add_span_data	add_span_data passed into bc_calc_len()
bc	bytecode containing span
id	non-zero identifier for span; may be any non-zero value if <0, expand is called for any change; if >0, expand is only called when exceeds threshold
value	dependent value for bytecode expansion
neg_thres	negative threshold for long/short decision
pos_thres	positive threshold for long/short decision

Definition at line 56 of file bytecode.h.

4.4.3.2 typedef struct yasm_bytecode_callback yasm_bytecode_callback

Bytecode callback structure.

Any implementation of a specific bytecode must implement these functions and this callback structure. The bytecode

implementation-specific data is stored in yasm_bytecode.contents.

4.4.3.3 typedef struct yasm_dataval yasm_dataval

A data value (opaque type).

Definition at line 38 of file bytecode.h.

4.4.3.4 typedef struct yasm_datavalhead yasm_datavalhead

A list of data values.

Definition at line 40 of file bytecode.h.

4.4.4 Function Documentation

4.4.4.1 STAILQ_HEAD (yasm_datavalhead , yasm_dataval)

Linked list of data values.

4.4.4.2 YASM_LIB_DECL void yasm_bc__add_symrec (yasm_bytecode * *bc,* yasm_symrec * *sym*)

Add to the list of symrecs that reference a bytecode.

For symrec use only.

Parameters

bc	bytecode
sym	symbol

4.4.4.3 YASM_LIB_DECL int yasm_bc_calc_len (yasm_bytecode * *bc,* yasm_bc_add_span_func *add_span,* void * *add_span_data*)

Resolve EQUs in a bytecode and calculate its minimum size.

Generates dependent bytecode spans for cases where, if the length spanned increases, it could cause the bytecode size to increase. Any bytecode multiple is NOT included in the length or spans generation; this must be handled at a higher level.

Parameters

bc	bytecode
add_span	function to call to add a span
add_span_data	extra data to be passed to add_span function

Returns

0 if no error occurred, nonzero if there was an error recognized (and output) during execution.

Note

May store to bytecode updated expressions and the short length.

4.4.4.4 YASM_LIB_DECL int yasm_bc_calc_len_common (yasm_bytecode * *bc,* yasm_bc_add_span_func *add_span,* void * *add_span_data*)

Common bytecode callback calc_len function, for where the bytecode has no calculatable length.

Causes an internal error if called.

4.4.4.5 YASM_LIB_DECL yasm_bytecode* yasm_bc_create_align (yasm_expr * *boundary,* yasm_expr * *fill,* yasm_expr * *maxskip,* const unsigned char ** *code_fill,* unsigned long *line*)

Create a bytecode that aligns the following bytecode to a boundary.

Parameters

boundary	byte alignment (must be a power of two)
fill	fill data (if NULL, code_fill or 0 is used)
maxskip	maximum number of bytes to skip
code_fill	code fill data (if NULL, 0 is used)
line	virtual line (from yasm_linemap)

Returns

Newly allocated bytecode.

Note

The precedence on generated fill is as follows:

- from fill parameter (if not NULL)
- from code_fill parameter (if not NULL)
- 0

4.4.4.6 YASM_LIB_DECL yasm_bytecode* yasm_bc_create_common (const yasm_bytecode_callback * *callback,* void * *contents,* unsigned long *line*)

Create a bytecode of any specified type.

Parameters

callback	bytecode callback functions, if NULL, creates empty bytecode (may not be resolved or output)
contents	type-specific data
line	virtual line (from yasm_linemap)

Returns

Newly allocated bytecode of the specified type.

4.4.4.7 YASM_LIB_DECL yasm_bytecode* yasm_bc_create_data (yasm_datavalhead * *datahead,* unsigned int *size,* int *append_zero,* yasm_arch * *arch,* unsigned long *line*)

Create a bytecode containing data value(s).

Parameters

datahead	list of data values (kept, do not free)
size	storage size (in bytes) for each data value
append_zero	append a single zero byte after each data value (if non-zero)
arch	architecture (optional); if provided, data items are directly simplified to bytes if possible
line	virtual line (from yasm_linemap)

Returns

Newly allocated bytecode.

4.4.4.8 YASM_LIB_DECL yasm_bytecode* yasm_bc_create_incbin (char * *filename,* yasm_expr * *start,* yasm_expr * *maxlen,* yasm_linemap * *linemap,* unsigned long *line*)

Create a bytecode that includes a binary file verbatim.

Parameters

filename	path to binary file (kept, do not free)
start	starting location in file (in bytes) to read data from (kept, do not free); may be NULL to indicate 0
maxlen	maximum number of bytes to read from the file (kept, do do not free); may be NULL to indicate no maximum
linemap	line mapping repository
line	virtual line (from yasm_linemap) for the bytecode

Returns

Newly allocated bytecode.

4.4.4.9 YASM_LIB_DECL yasm_bytecode* yasm_bc_create_leb128 (yasm_datavalhead * *datahead,* int *sign,* unsigned long *line*)

Create a bytecode containing LEB128-encoded data value(s).

Parameters

datahead	list of data values (kept, do not free)
sign	signedness (1=signed, 0=unsigned) of each data value
line	virtual line (from yasm_linemap)

Returns

Newly allocated bytecode.

4.4.4.10 YASM_LIB_DECL yasm_bytecode* yasm_bc_create_org (unsigned long *start,* unsigned long *fill,* unsigned long *line*)

Create a bytecode that puts the following bytecode at a fixed section offset.

Parameters

start	section offset of following bytecode
fill	fill value
line	virtual line (from yasm_linemap)

Returns

Newly allocated bytecode.

4.4.4.11 YASM_LIB_DECL yasm_bytecode* yasm_bc_create_reserve (yasm_expr * *numitems*, unsigned int *itemsize*, unsigned long *line*)

Create a bytecode reserving space.

Parameters

numitems	number of reserve "items" (kept, do not free)
itemsize	reserved size (in bytes) for each item
line	virtual line (from yasm_linemap)

Returns

Newly allocated bytecode.

4.4.4.12 YASM_LIB_DECL void yasm_bc_destroy (yasm_bytecode * *bc*)

Delete (free allocated memory for) a bytecode.

Parameters

bc	bytecode (only pointer to it); may be NULL

4.4.4.13 YASM_LIB_DECL int yasm_bc_elem_size (yasm_bytecode * *bc*)

Return elemens size of a data bytecode.

Returns the size of each elements of a data bytecode, for proper dereference of symbols attached to it.

Parameters

bc	bytecode

Returns

0 if element size is unknown

4.4.4.14 YASM_LIB_DECL int yasm_bc_expand (yasm_bytecode * *bc*, int *span*, long *old_val*, long *new_val*, long * *neg_thres*, long * *pos_thres*)

Recalculate a bytecode's length based on an expanded span length.

Parameters

bc	bytecode
span	span ID (as given to yasm_bc_add_span_func in yasm_bc_calc_len)
old_val	previous span value
new_val	new span value
neg_thres	negative threshold for long/short decision (returned)
pos_thres	positive threshold for long/short decision (returned)

Returns

0 if bc no longer dependent on this span's length, negative if there was an error recognized (and output) during execution, and positive if bc size may increase for this span further based on the new negative and positive thresholds returned.

Note

May store to bytecode updated expressions and the updated length.

4.4.4.15 YASM_LIB_DECL int yasm_bc_expand_common (yasm_bytecode * *bc*, int *span*, long *old_val*, long *new_val*, long * *neg_thres*, long * *pos_thres*)

Common bytecode callback expand function, for where the bytecode is always short (calc_len never calls add_span).

Causes an internal error if called.

4.4.4.16 YASM_LIB_DECL void yasm_bc_finalize (yasm_bytecode * *bc*, yasm_bytecode * *prev_bc*)

Finalize a bytecode after parsing.

Parameters

bc	bytecode
prev_bc	bytecode directly preceding bc in a list of bytecodes

4.4.4.17 YASM_LIB_DECL yasm_insn* yasm_bc_get_insn (yasm_bytecode * *bc*)

Get a yasm_insn structure from an instruction bytecode (if possible).

Parameters

bc	bytecode

Returns

Instruction details if bytecode is an instruction bytecode, otherwise NULL.

4.4.4.18 YASM_LIB_DECL int yasm_bc_get_multiple (yasm_bytecode * *bc*, long * *multiple*, int *calc_bc_dist*)

Get the bytecode multiple value as an integer.

Parameters

bc	bytecode
multiple	multiple value (output)
calc_bc_dist	nonzero if distances between bytecodes should be calculated, 0 if error should be returned in this case

Returns

1 on error (set with yasm_error_set), 0 on success.

4.4.4.19 YASM_LIB_DECL const yasm_expr* yasm_bc_get_multiple_expr (const yasm_bytecode * *bc*)

Get the bytecode multiple value as an expression.

Parameters

bc	bytecode

Returns

Bytecode multiple, NULL if =1.

4.4.4.20 YASM_LIB_DECL yasm_section* yasm_bc_get_section (yasm_bytecode * *bc*)

Get the section that contains a particular bytecode.

Parameters

bc	bytecode

Returns

Section containing bc (can be NULL if bytecode is not part of a section).

4.4.4.21 YASM_LIB_DECL unsigned long yasm_bc_next_offset (yasm_bytecode * *precbc*)

Get the offset of the next bytecode (the next bytecode doesn't have to actually exist).

Parameters

precbc	preceding bytecode

Returns

Offset of the next bytecode in bytes.

Warning

Only valid /after/ optimization.

4.4.4.22 YASM_LIB_DECL void yasm_bc_print (const yasm_bytecode * *bc*, FILE * *f*, int *indent_level*)

Print a bytecode.

For debugging purposes.

Parameters

f	file
indent_level	indentation level
bc	bytecode

4.4.4.23 YASM_LIB_DECL const yasm_expr* yasm_bc_reserve_numitems (yasm_bytecode * *bc*, unsigned int * *itemsize*)

Get the number of items and itemsize for a reserve bytecode.

If bc is not a reserve bytecode, returns NULL.

Parameters

bc	bytecode
itemsize	reserved size (in bytes) for each item (returned)

Returns

NULL if bc is not a reserve bytecode, otherwise an expression for the number of items to reserve.

4.4.4.24 YASM_LIB_DECL void yasm_bc_set_multiple (yasm_bytecode * *bc*, yasm_expr * *e*)

Set multiple field of a bytecode.

A bytecode can be repeated a number of times when output. This function sets that multiple.

Parameters

bc	bytecode
e	multiple (kept, do not free)

4.4.4.25 YASM_LIB_DECL unsigned char* yasm_bc_tobytes (yasm_bytecode * *bc*, unsigned char * *buf*, unsigned long * *bufsize*, int * *gap*, void * *d*, yasm_output_value_func *output_value*, yasm_output_reloc_func *output_reloc*)

Convert a bytecode into its byte representation.

Parameters

bc	bytecode
buf	byte representation destination buffer
bufsize	size of buf (in bytes) prior to call; size of the generated data after call
gap	if nonzero, indicates the data does not really need to exist in the object file; if nonzero, contents of buf are undefined [output]
d	data to pass to each call to output_value/output_reloc
output_value	function to call to convert values into their byte representation
output_reloc	function to call to output relocation entries for a single sym

Returns

Newly allocated buffer that should be used instead of buf for reading the byte representation, or NULL if buf was big enough to hold the entire byte representation.

Note

Calling twice on the same bytecode may *not* produce the same results on the second call, as calling this function may result in non-reversible changes to the bytecode.

4.4.4.26 YASM_LIB_DECL int yasm_bc_tobytes_common (yasm_bytecode * *bc*, unsigned char ** *bufp*, unsigned char * *bufstart*, void * *d*, yasm_output_value_func *output_value*, yasm_output_reloc_func *output_reloc*)

Common bytecode callback tobytes function, for where the bytecode cannot be converted to bytes.

Causes an internal error if called.

4.4.4.27 YASM_LIB_DECL void yasm_bc_transform (yasm_bytecode * *bc*, const yasm_bytecode_callback * *callback*, void * *contents*)

Transform a bytecode of any type into a different type.

Parameters

bc	bytecode to transform
callback	new bytecode callback function
contents	new type-specific data

4.4.4.28 YASM_LIB_DECL yasm_intnum* yasm_calc_bc_dist (yasm_bytecode * *precbc1*, yasm_bytecode * *precbc2*)

Determine the distance between the starting offsets of two bytecodes.

Parameters

precbc1	preceding bytecode to the first bytecode
precbc2	preceding bytecode to the second bytecode

Returns

Distance in bytes between the two bytecodes (bc2-bc1), or NULL if the distance was indeterminate.

Warning

Only valid /after/ optimization.

4.4.4.29 YASM_LIB_DECL yasm_dataval* yasm_dv_create_expr (yasm_expr * *expn*)

Create a new data value from an expression.

Parameters

expn	expression

Returns

Newly allocated data value.

4.4.4.30 YASM_LIB_DECL yasm_dataval* yasm_dv_create_raw (unsigned char * *contents*, unsigned long *len*)

Create a new data value from raw bytes data.

Parameters

contents	raw data (may contain NULs)
len	length

Returns

Newly allocated data value.

4.4.4.31 YASM_LIB_DECL yasm_dataval* yasm_dv_create_reserve (void)

Create a new uninitialized data value.

Returns

Newly allocated data value.

4.4.4.32 YASM_LIB_DECL yasm_dataval* yasm_dv_create_string (char * *contents*, size_t *len*)

Create a new data value from a string.

Parameters

contents	string (may contain NULs)
len	length of string

Returns

Newly allocated data value.

4.4.4.33 YASM_LIB_DECL int yasm_dv_get_multiple (yasm_dataval * *dv*, unsigned long * *multiple*)

Get the data value multiple value as an unsigned long integer.

Parameters

dv	data value
multiple	multiple value (output)

Returns

1 on error (set with yasm_error_set), 0 on success.

4.4.4.34 YASM_LIB_DECL yasm_value* yasm_dv_get_value (yasm_dataval * *dv*)

Get the underlying value of a data value.

Parameters

dv	data value

Returns

Value, or null if non-value (e.g. string or raw).

4.4.4.35 YASM_LIB_DECL void yasm_dv_set_multiple (yasm_dataval * *dv*, yasm_expr * *e*)

Set multiple field of a data value.

A data value can be repeated a number of times when output. This function sets that multiple.

Parameters

dv	data value
e	multiple (kept, do not free)

4.4.4.36 YASM_LIB_DECL yasm_dataval* yasm_dvs_append (yasm_datavalhead * *headp*, yasm_dataval * *dv*)

Add data value to the end of a list of data values.

Note

Does not make a copy of the data value; so don't pass this function static or local variables, and discard the dv pointer after calling this function.

Parameters

headp	data value list
dv	data value (may be NULL)

Returns

If data value was actually appended (it wasn't NULL), the data value; otherwise NULL.

4.4.4.37 YASM_LIB_DECL void yasm_dvs_delete (yasm_datavalhead * *headp*)

Delete (free allocated memory for) a list of data values.

Parameters

headp	list of data values

4.4.4.38 void yasm_dvs_initialize (yasm_datavalhead * *headp*)

Initialize a list of data values.

Parameters

headp	list of data values

4.4.4.39 YASM_LIB_DECL void yasm_dvs_print (const yasm_datavalhead * *headp*, FILE * *f*, int *indent_level*)

Print a data value list.

For debugging purposes.

Parameters

f	file
indent_level	indentation level
headp	data value list

4.5 libyasm/coretype.h File Reference

YASM core types and utility functions.

Data Structures

- struct yasm_stdmac

  Standard macro structure for modules that allows association of a set of standard macros with a parser/preprocessor combination.
- struct yasm_assoc_data_callback

  YASM associated data callback structure.
- struct yasm_value

  A value.

Macros

- #define **YASM_LIB_DECL**
- #define YASM_VALUE_RSHIFT_MAX 127

  Maximum value of yasm_value.rshift.

Typedefs

- typedef struct yasm_arch yasm_arch

  Architecture instance (mostly opaque type).
- typedef struct yasm_preproc yasm_preproc

  Preprocessor interface.
- typedef struct yasm_parser yasm_parser

  Parser instance (mostly opaque type).
- typedef struct yasm_objfmt yasm_objfmt

  Object format interface.
- typedef struct yasm_dbgfmt yasm_dbgfmt

  Debug format interface.
- typedef struct yasm_listfmt yasm_listfmt

  List format interface.
- typedef struct yasm_objfmt_module yasm_objfmt_module

  Object format module interface.
- typedef struct yasm_dbgfmt_module yasm_dbgfmt_module

  Debug format module interface.
- typedef struct yasm_stdmac yasm_stdmac

  Standard macro structure for modules that allows association of a set of standard macros with a parser/preprocessor combination.
- typedef struct yasm_assoc_data_callback yasm_assoc_data_callback

  YASM associated data callback structure.
- typedef struct yasm_errwarns yasm_errwarns

  Set of collected error/warnings (opaque type).
- typedef struct yasm_bytecode yasm_bytecode

Bytecode.

- typedef struct yasm_object yasm_object

  Object.
- typedef struct yasm_section yasm_section

  Section (opaque type).
- typedef struct yasm_symtab yasm_symtab

  Symbol table (opaque type).
- typedef struct yasm_symrec yasm_symrec

  Symbol record (opaque type).
- typedef struct yasm_expr yasm_expr

  Expression.
- typedef struct yasm_intnum yasm_intnum

  Integer value (opaque type).
- typedef struct yasm_floatnum yasm_floatnum

  Floating point value (opaque type).
- typedef struct yasm_value yasm_value

  A value.
- typedef struct yasm_linemap yasm_linemap

  Line number mapping repository (opaque type).
- typedef struct yasm_valparam yasm_valparam

  Value/parameter pair (opaque type).
- typedef struct yasm_valparamhead yasm_valparamhead

  List of value/parameters (opaque type).
- typedef struct yasm_directive yasm_directive

  Directive list entry.
- typedef struct yasm_effaddr yasm_effaddr

  An effective address.
- typedef struct yasm_insn yasm_insn

  An instruction.
- typedef enum yasm_expr_op yasm_expr_op

  Expression operators usable in yasm_expr expressions.
- typedef int(∗ yasm_output_value_func)(yasm_value ∗value, unsigned char ∗buf, unsigned int destsize, unsigned long offset, yasm_bytecode ∗bc, int warn, void ∗d)

  Convert yasm_value to its byte representation.
- typedef int(∗ yasm_output_reloc_func)(yasm_symrec ∗sym, yasm_bytecode ∗bc, unsigned char ∗buf, unsigned int destsize, unsigned int valsize, int warn, void ∗d)

  Convert a symbol reference to its byte representation.

Enumerations

- enum yasm_expr_op {
  YASM_EXPR_IDENT, YASM_EXPR_ADD, YASM_EXPR_SUB, YASM_EXPR_MUL,
  YASM_EXPR_DIV, YASM_EXPR_SIGNDIV, YASM_EXPR_MOD, YASM_EXPR_SIGNMOD,
  YASM_EXPR_NEG, YASM_EXPR_NOT, YASM_EXPR_OR, YASM_EXPR_AND,
  YASM_EXPR_XOR, YASM_EXPR_XNOR, YASM_EXPR_NOR, YASM_EXPR_SHL,
  YASM_EXPR_SHR, YASM_EXPR_LOGIC, YASM_EXPR_LOR, YASM_EXPR_LAND,
  YASM_EXPR_LNOT, YASM_EXPR_LXOR, YASM_EXPR_LXNOR, YASM_EXPR_LNOR,
  YASM_EXPR_LT, YASM_EXPR_GT, YASM_EXPR_EQ, YASM_EXPR_LE,
  YASM_EXPR_GE, YASM_EXPR_NE, YASM_EXPR_NONNUM, YASM_EXPR_SEG,
  YASM_EXPR_WRT, YASM_EXPR_SEGOFF }

Expression operators usable in yasm_expr expressions.

Functions

- YASM_LIB_DECL int yasm__mergesort (void *base, size_t nmemb, size_t size, int(*compar)(const void *, const void *))

  Sort an array using merge sort algorithm.
- YASM_LIB_DECL char * yasm__strsep (char **stringp, const char *delim)

  Separate string by delimiters.
- YASM_LIB_DECL int yasm__strcasecmp (const char *s1, const char *s2)

  Compare two strings, ignoring case differences.
- YASM_LIB_DECL int yasm__strncasecmp (const char *s1, const char *s2, size_t n)

  Compare portion of two strings, ignoring case differences.
- YASM_LIB_DECL char * yasm__xstrdup (const char *str)

  strdup() implementation using yasm_xmalloc().
- YASM_LIB_DECL char * yasm__xstrndup (const char *str, size_t max)

  strndup() implementation using yasm_xmalloc().

Variables

- YASM_LIB_DECL void *(* yasm_xmalloc)(size_t size)

  Error-checking memory allocation.
- YASM_LIB_DECL void *(* yasm_xcalloc)(size_t nelem, size_t elsize)

  Error-checking memory allocation (with clear-to-0).
- YASM_LIB_DECL void *(* yasm_xrealloc)(void *oldmem, size_t size)

  Error-checking memory reallocation.
- YASM_LIB_DECL void(* yasm_xfree)(void *p)

  Error-checking memory deallocation.

4.5.1 Detailed Description

YASM core types and utility functions.

Definition in file coretype.h.

4.5.2 Typedef Documentation

4.5.2.1 typedef struct yasm_arch yasm_arch

Architecture instance (mostly opaque type).

See Also

arch.h for details.

Definition at line 38 of file coretype.h.

4.5.2.2 typedef struct yasm_assoc_data_callback yasm_assoc_data_callback

YASM associated data callback structure.

Many data structures can have arbitrary data associated with them.

4.5.2.3 typedef struct yasm_bytecode yasm_bytecode

Bytecode.

See Also

bytecode.h for details and related functions.

Definition at line 93 of file coretype.h.

4.5.2.4 typedef struct yasm_dbgfmt yasm_dbgfmt

Debug format interface.

See Also

dbgfmt.h for details.

Definition at line 46 of file coretype.h.

4.5.2.5 typedef struct yasm_dbgfmt_module yasm_dbgfmt_module

Debug format module interface.

See Also

dbgfmt.h for details.

Definition at line 53 of file coretype.h.

4.5.2.6 typedef struct yasm_directive yasm_directive

Directive list entry.

See Also

valparam.h for details and related functions.

Definition at line 205 of file coretype.h.

4.5.2.7 typedef struct yasm_effaddr yasm_effaddr

An effective address.

See Also

insn.h for related functions.

Definition at line 210 of file coretype.h.

4.5.2.8 typedef struct yasm_errwarns yasm_errwarns

Set of collected error/warnings (opaque type).

See Also

errwarn.h for details.

Definition at line 90 of file coretype.h.

4.5.2.9 typedef struct yasm_expr yasm_expr

Expression.

See Also

expr.h for details and related functions.

Definition at line 108 of file coretype.h.

4.5.2.10 typedef enum yasm_expr_op yasm_expr_op

Expression operators usable in yasm_expr expressions.

4.5.2.11 typedef struct yasm_floatnum yasm_floatnum

Floating point value (opaque type).

See Also

floatnum.h for related functions.

Definition at line 114 of file coretype.h.

4.5.2.12 typedef struct yasm_insn yasm_insn

An instruction.

See Also

insn.h for related functions.

Definition at line 215 of file coretype.h.

4.5.2.13 typedef struct yasm_intnum yasm_intnum

Integer value (opaque type).

See Also

intnum.h for related functions.

Definition at line 110 of file coretype.h.

4.5.2.14 typedef struct yasm_linemap yasm_linemap

Line number mapping repository (opaque type).

See Also

linemap.h for related functions.

Definition at line 192 of file coretype.h.

4.5.2.15 typedef struct yasm_listfmt yasm_listfmt

List format interface.

See Also

listfmt.h for details.

Definition at line 48 of file coretype.h.

4.5.2.16 typedef struct yasm_object yasm_object

Object.

See Also

section.h for details and related functions.

Definition at line 96 of file coretype.h.

4.5.2.17 typedef struct yasm_objfmt yasm_objfmt

Object format interface.

See Also

objfmt.h for details.

Definition at line 44 of file coretype.h.

4.5.2.18 typedef struct yasm_objfmt_module yasm_objfmt_module

Object format module interface.

See Also

objfmt.h for details.

Definition at line 51 of file coretype.h.

4.5.2.19 typedef int(∗ yasm_output_reloc_func)(yasm_symrec ∗sym, yasm_bytecode ∗bc, unsigned char ∗buf, unsigned int destsize, unsigned int valsize, int warn, void ∗d)

Convert a symbol reference to its byte representation.

Usually implemented by object formats and debug formats to keep track of relocations generated by themselves.

Parameters

sym	symbol
bc	current bytecode (usually passed into higher-level calling function)
buf	buffer for byte representation

destsize	destination size (in bytes)
valsize	size (in bits)
warn	enables standard warnings: zero for none; nonzero for overflow/underflow floating point warnings; negative for signed integer warnings, positive for unsigned integer warnings
d	objfmt-specific data (passed into higher-level calling function)

Returns

Nonzero if an error occurred, 0 otherwise.

Definition at line 296 of file coretype.h.

4.5.2.20 typedef int(∗ yasm_output_value_func)(yasm_value ∗value,unsigned char ∗buf, unsigned int destsize, unsigned long offset, yasm_bytecode ∗bc, int warn,void ∗d)

Convert yasm_value to its byte representation.

Usually implemented by object formats to keep track of relocations and verify legal expressions. Must put the value into the least significant bits of the destination, unless shifted into more significant bits by the shift parameter. The destination bits must be cleared before being set.

Parameters

value	value
buf	buffer for byte representation
destsize	destination size (in bytes)
offset	offset (in bytes) of the expr contents from the start of the bytecode (needed for relative)
bc	current bytecode (usually passed into higher-level calling function)
warn	enables standard warnings: zero for none; nonzero for overflow/underflow floating point warnings
d	objfmt-specific data (passed into higher-level calling function)

Returns

Nonzero if an error occurred, 0 otherwise.

Definition at line 275 of file coretype.h.

4.5.2.21 typedef struct yasm_parser yasm_parser

Parser instance (mostly opaque type).

See Also

parser.h for details.

Definition at line 42 of file coretype.h.

4.5.2.22 typedef struct yasm_preproc yasm_preproc

Preprocessor interface.

See Also

preproc.h for details.

Definition at line 40 of file coretype.h.

4.5.2.23 typedef struct yasm_section yasm_section

Section (opaque type).

See Also

section.h for related functions.

Definition at line 99 of file coretype.h.

4.5.2.24 typedef struct yasm_stdmac yasm_stdmac

Standard macro structure for modules that allows association of a set of standard macros with a parser/preprocessor combination.

A NULL-terminated array of these structures is used in a number of module interfaces.

4.5.2.25 typedef struct yasm_symrec yasm_symrec

Symbol record (opaque type).

See Also

symrec.h for related functions.

Definition at line 105 of file coretype.h.

4.5.2.26 typedef struct yasm_symtab yasm_symtab

Symbol table (opaque type).

See Also

symrec.h for related functions.

Definition at line 102 of file coretype.h.

4.5.2.27 typedef struct yasm_valparam yasm_valparam

Value/parameter pair (opaque type).

See Also

valparam.h for related functions.

Definition at line 197 of file coretype.h.

4.5.2.28 typedef struct yasm_valparamhead yasm_valparamhead

List of value/parameters (opaque type).

See Also

valparam.h for related functions.

Definition at line 201 of file coretype.h.

4.5.2.29 typedef struct yasm_value yasm_value

A value.

May be absolute or relative. Outside the parser, yasm_expr should only be used for absolute exprs. Anything that could contain a relocatable value should use this structure instead.

See Also

value.h for related functions.

4.5.3 Enumeration Type Documentation

4.5.3.1 enum yasm_expr_op

Expression operators usable in yasm_expr expressions.

Enumerator

YASM_EXPR_IDENT No operation, just a value.
YASM_EXPR_ADD Arithmetic addition (+).
YASM_EXPR_SUB Arithmetic subtraction (-).
YASM_EXPR_MUL Arithmetic multiplication (*).
YASM_EXPR_DIV Arithmetic unsigned division.
YASM_EXPR_SIGNDIV Arithmetic signed division.
YASM_EXPR_MOD Arithmetic unsigned modulus.
YASM_EXPR_SIGNMOD Arithmetic signed modulus.
YASM_EXPR_NEG Arithmetic negation (-).
YASM_EXPR_NOT Bitwise negation.
YASM_EXPR_OR Bitwise OR.
YASM_EXPR_AND Bitwise AND.
YASM_EXPR_XOR Bitwise XOR.
YASM_EXPR_XNOR Bitwise XNOR.
YASM_EXPR_NOR Bitwise NOR.
YASM_EXPR_SHL Shift left (logical).
YASM_EXPR_SHR Shift right (logical).
YASM_EXPR_LOGIC Start of logic operations (not an op).
YASM_EXPR_LOR Logical OR.
YASM_EXPR_LAND Logical AND.
YASM_EXPR_LNOT Logical negation.
YASM_EXPR_LXOR Logical XOR.
YASM_EXPR_LXNOR Logical XNOR.
YASM_EXPR_LNOR Logical NOR.
YASM_EXPR_LT Less than comparison.
YASM_EXPR_GT Greater than comparison.
YASM_EXPR_EQ Equality comparison.
YASM_EXPR_LE Less than or equal to comparison.

YASM_EXPR_GE Greater than or equal to comparison.

YASM_EXPR_NE Not equal comparison.

YASM_EXPR_NONNUM Start of non-numeric operations (not an op).

YASM_EXPR_SEG SEG operator (gets segment portion of address).

YASM_EXPR_WRT WRT operator (gets offset of address relative to some other segment).

YASM_EXPR_SEGOFF The ':' in segment:offset.

Definition at line 218 of file coretype.h.

4.5.4 Function Documentation

4.5.4.1 YASM_LIB_DECL int yasm__mergesort (void * *base*, size_t *nmemb*, size_t *size*, int(*)(const void *, const void *) *compar*)

Sort an array using merge sort algorithm.

Parameters

base	base of array
nmemb	number of elements in array
size	size of each array element
compar	element comparison function

4.5.4.2 YASM_LIB_DECL int yasm__strcasecmp (const char * *s1*, const char * *s2*)

Compare two strings, ignoring case differences.

Parameters

s1	string 1
s2	string 2

Returns

0 if strings are equal, -1 if s1<s2, 1 if s1>s2.

4.5.4.3 YASM_LIB_DECL int yasm__strncasecmp (const char * *s1*, const char * *s2*, size_t *n*)

Compare portion of two strings, ignoring case differences.

Parameters

s1	string 1
s2	string 2
n	maximum number of characters to compare

Returns

0 if strings are equal, -1 if s1<s2, 1 if s1>s2.

4.5.4.4 YASM_LIB_DECL char* yasm__strsep (char ** *stringp,* const char * *delim*)

Separate string by delimiters.

Parameters

stringp	string
delim	set of 1 or more delimiters

Returns

First/next substring.

4.5.4.5 YASM_LIB_DECL char* yasm__xstrdup (const char * *str*)

strdup() implementation using yasm_xmalloc().

Parameters

str	string

Returns

Newly allocated duplicate string.

4.5.4.6 YASM_LIB_DECL char* yasm__xstrndup (const char * *str,* size_t *max*)

strndup() implementation using yasm_xmalloc().

Parameters

str	string
max	maximum number of characters to copy

Returns

Newly allocated duplicate string.

4.5.5 Variable Documentation

4.5.5.1 YASM_LIB_DECL void*(* yasm_xcalloc)(size_t nelem, size_t elsize)

Error-checking memory allocation (with clear-to-0).

A default implementation is provided that calls yasm_fatal() on allocation errors. A replacement should *never* return NULL.

Parameters

size	number of elements to allocate
elsize	size (in bytes) of each element

Returns

Allocated and cleared memory block.

4.5.5.2 YASM_LIB_DECL void(* yasm_xfree)(void *p)

Error-checking memory deallocation.

A default implementation is provided that calls yasm_fatal() on allocation errors.

Parameters

p	memory block to free

4.5.5.3 YASM_LIB_DECL void*(* yasm_xmalloc)(size_t size)

Error-checking memory allocation.

A default implementation is provided that calls yasm_fatal() on allocation errors. A replacement should *never* return NULL.

Parameters

size	number of bytes to allocate

Returns

Allocated memory block.

4.5.5.4 YASM_LIB_DECL void*(* yasm_xrealloc)(void *oldmem, size_t size)

Error-checking memory reallocation.

A default implementation is provided that calls yasm_fatal() on allocation errors. A replacement should *never* return NULL.

Parameters

oldmem	memory block to resize
elsize	new size, in bytes

Returns

Re-allocated memory block.

4.6 libyasm/dbgfmt.h File Reference

YASM debug format interface.

Data Structures

- struct yasm_dbgfmt_module

  Debug format module interface.

Functions

- const char * yasm_dbgfmt_keyword (const yasm_dbgfmt *dbgfmt)

  Get the keyword used to select a debug format.
- yasm_dbgfmt * yasm_dbgfmt_create (const yasm_dbgfmt_module *module, yasm_object *object)

  Initialize debug output for use.
- void yasm_dbgfmt_destroy (yasm_dbgfmt *dbgfmt)

  Cleans up any allocated debug format memory.
- void yasm_dbgfmt_generate (yasm_object *object, yasm_linemap *linemap, yasm_errwarns *errwarns)

  Generate debugging information bytecodes.

4.6.1 Detailed Description

YASM debug format interface.

Definition in file dbgfmt.h.

4.6.2 Function Documentation

4.6.2.1 yasm_dbgfmt* yasm_dbgfmt_create (const yasm_dbgfmt_module * *module*, yasm_object * *object*)

Initialize debug output for use.

Must call before any other debug format functions. The filenames are provided solely for informational purposes.

Parameters

module	debug format module
object	object to generate debugging information for

Returns

NULL if object format does not provide needed support.

4.6.2.2 void yasm_dbgfmt_destroy (yasm_dbgfmt * *dbgfmt*)

Cleans up any allocated debug format memory.

Parameters

dbgfmt	debug format

4.6.2.3 void yasm_dbgfmt_generate (yasm_object * *object*, yasm_linemap * *linemap*, yasm_errwarns * *errwarns*)

Generate debugging information bytecodes.

Parameters

object	object
linemap	virtual/physical line mapping
errwarns	error/warning set

Note

Errors and warnings are stored into errwarns.

4.6.2.4 const char* yasm_dbgfmt_keyword (const yasm_dbgfmt * *dbgfmt*)

Get the keyword used to select a debug format.

Parameters

dbgfmt	debug format

Returns

keyword

4.7 libyasm/errwarn.h File Reference

YASM error and warning reporting interface.

Macros

- #define **YASM_LIB_DECL**
- #define yasm_internal_error(message) yasm_internal_error_(__FILE__, __LINE__, message)

  Easily-callable version of yasm_internal_error_().

Typedefs

- typedef enum yasm_warn_class yasm_warn_class

  Warning classes (that may be enabled/disabled).
- typedef enum yasm_error_class yasm_error_class

  Error classes.
- typedef void(* yasm_print_error_func)(const char *fn, unsigned long line, const char *msg, const char *xref_fn, unsigned long xref_line, const char *xref_msg)

  Print out an error.
- typedef void(* yasm_print_warning_func)(const char *fn, unsigned long line, const char *msg)

  Print out a warning.

Enumerations

- enum yasm_warn_class {
  YASM_WARN_NONE = 0, YASM_WARN_GENERAL, YASM_WARN_UNREC_CHAR, YASM_WARN_PREPROC,
  YASM_WARN_ORPHAN_LABEL, YASM_WARN_UNINIT_CONTENTS, YASM_WARN_SIZE_OVERRIDE, YASM_WARN_IMPLICIT_SIZE_OVERRIDE }

  Warning classes (that may be enabled/disabled).
- enum yasm_error_class {
  YASM_ERROR_NONE = 0x0000, YASM_ERROR_GENERAL = 0xFFFF, YASM_ERROR_ARITHMETIC =

0x0001, YASM_ERROR_OVERFLOW = 0x8001,
YASM_ERROR_FLOATING_POINT = 0x4001, YASM_ERROR_ZERO_DIVISION = 0x2001, YASM_ERROR_-ASSERTION = 0x0002, YASM_ERROR_VALUE = 0x0004,
YASM_ERROR_NOT_ABSOLUTE = 0x8004, YASM_ERROR_TOO_COMPLEX = 0x4004, YASM_ERROR_N-OT_CONSTANT = 0x2004, YASM_ERROR_IO = 0x0008,
YASM_ERROR_NOT_IMPLEMENTED = 0x0010, YASM_ERROR_TYPE = 0x0020, YASM_ERROR_SYNTAX = 0x0040, YASM_ERROR_PARSE = 0x8040 }

Error classes.

Functions

- YASM_LIB_DECL void yasm_errwarn_initialize (void)

  Initialize any internal data structures.
- YASM_LIB_DECL void yasm_errwarn_cleanup (void)

  Clean up any memory allocated by yasm_errwarn_initialize() or other functions.
- YASM_LIB_DECL void yasm__fatal (const char *message,...)

  Reporting point of fatal errors, with variable arguments (internal only).
- YASM_LIB_DECL void yasm_error_clear (void)

  Unconditionally clear the error indicator, freeing any associated data.
- yasm_error_class yasm_error_occurred (void)

  Get the error indicator.
- YASM_LIB_DECL int yasm_error_matches (yasm_error_class eclass)

  Check the error indicator against an error class.
- YASM_LIB_DECL void yasm_error_set_va (yasm_error_class eclass, const char *format, va_list va)

  Set the error indicator (va_list version).
- YASM_LIB_DECL void yasm_error_set (yasm_error_class eclass, const char *format,...)

  Set the error indicator.
- YASM_LIB_DECL void yasm_error_set_xref_va (unsigned long xrefline, const char *format, va_list va)

  Set a cross-reference for a new error (va_list version).
- YASM_LIB_DECL void yasm_error_set_xref (unsigned long xrefline, const char *format,...)

  Set a cross-reference for a new error.
- YASM_LIB_DECL void yasm_error_fetch (yasm_error_class *eclass, char **str, unsigned long *xrefline, char **xrefstr)

  Fetch the error indicator and all associated data.
- YASM_LIB_DECL void yasm_warn_clear (void)

  Unconditionally clear all warning indicators, freeing any associated data.
- YASM_LIB_DECL yasm_warn_class yasm_warn_occurred (void)

  Get the first warning indicator.
- YASM_LIB_DECL void yasm_warn_set_va (yasm_warn_class wclass, const char *format, va_list va)

  Add a warning indicator (va_list version).
- YASM_LIB_DECL void yasm_warn_set (yasm_warn_class wclass, const char *format,...)

  Add a warning indicator.
- YASM_LIB_DECL void yasm_warn_fetch (yasm_warn_class *wclass, char **str)

  Fetch the first warning indicator and all associated data.
- YASM_LIB_DECL void yasm_warn_enable (yasm_warn_class wclass)

  Enable a class of warnings.
- YASM_LIB_DECL void yasm_warn_disable (yasm_warn_class wclass)

  Disable a class of warnings.

- YASM_LIB_DECL void yasm_warn_disable_all (void)

  Disable all classes of warnings.
- YASM_LIB_DECL yasm_errwarns * yasm_errwarns_create (void)

  Create an error/warning set for collection of multiple error/warnings.
- YASM_LIB_DECL void yasm_errwarns_destroy (yasm_errwarns *errwarns)

  Destroy an error/warning set.
- YASM_LIB_DECL void yasm_errwarn_propagate (yasm_errwarns *errwarns, unsigned long line)

  Propagate error indicator and warning indicator(s) to an error/warning set.
- YASM_LIB_DECL unsigned int yasm_errwarns_num_errors (yasm_errwarns *errwarns, int warning_as_error)

  Get total number of errors logged.
- YASM_LIB_DECL void yasm_errwarns_output_all (yasm_errwarns *errwarns, yasm_linemap *lm, int warning_as_error, yasm_print_error_func print_error, yasm_print_warning_func print_warning)

  Outputs error/warning set in sorted order (sorted by virtual line number).
- YASM_LIB_DECL char * yasm__conv_unprint (int ch)

  Convert a possibly unprintable character into a printable string.

Variables

- YASM_LIB_DECL void(* yasm_internal_error_)(const char *file, unsigned int line, const char *message)

  Reporting point of internal errors.
- YASM_LIB_DECL void(* yasm_fatal)(const char *message, va_list va)

  Reporting point of fatal errors.
- YASM_LIB_DECL const char *(* yasm_gettext_hook)(const char *msgid)

  Hook for library users to map to gettext() if GNU gettext is being used.

4.7.1 Detailed Description

YASM error and warning reporting interface.

Definition in file errwarn.h.

4.7.2 Macro Definition Documentation

4.7.2.1 #define yasm_internal_error(*message*) yasm_internal_error_(__FILE__, __LINE__, message)

Easily-callable version of yasm_internal_error_().

Automatically uses **FILE** and **LINE** as the file and line.

Parameters

message	internal error message

Definition at line 96 of file errwarn.h.

4.7.3 Typedef Documentation

4.7.3.1 typedef enum yasm_error_class yasm_error_class

Error classes.

Bitmask-based to support limited subclassing.

4.7.3.2 typedef void(∗ yasm_print_error_func)(const char ∗fn, unsigned long line, const char ∗msg,const char ∗xref_fn, unsigned long xref_line,const char ∗xref_msg)

Print out an error.

Parameters

fn	filename of source file
line	line number
msg	error message
xref_fn	cross-referenced source filename
xref_line	cross-referenced line number
xref_msg	cross-referenced error message

Definition at line 309 of file errwarn.h.

4.7.3.3 typedef void(∗ yasm_print_warning_func)(const char ∗fn, unsigned long line, const char ∗msg)

Print out a warning.

Parameters

fn	filename of source file
line	line number
msg	warning message

Definition at line 319 of file errwarn.h.

4.7.3.4 typedef enum yasm_warn_class yasm_warn_class

Warning classes (that may be enabled/disabled).

4.7.4 Enumeration Type Documentation

4.7.4.1 enum yasm_error_class

Error classes.

Bitmask-based to support limited subclassing.

Enumerator

YASM_ERROR_NONE No error.

YASM_ERROR_GENERAL Non-specific.

YASM_ERROR_ARITHMETIC Arithmetic error (general)

YASM_ERROR_OVERFLOW Arithmetic overflow.

YASM_ERROR_FLOATING_POINT Floating point error.

YASM_ERROR_ZERO_DIVISION Divide-by-zero.

YASM_ERROR_ASSERTION Assertion error.

YASM_ERROR_VALUE Value inappropriate (e.g. not in range)

YASM_ERROR_NOT_ABSOLUTE Absolute expression required.

YASM_ERROR_TOO_COMPLEX Expression too complex.

YASM_ERROR_NOT_CONSTANT Constant expression required.

YASM_ERROR_IO I/O error.

YASM_ERROR_NOT_IMPLEMENTED Not implemented error.

YASM_ERROR_TYPE Type error.

YASM_ERROR_SYNTAX Syntax error.

YASM_ERROR_PARSE Parser error.

Definition at line 50 of file errwarn.h.

4.7.4.2 enum yasm_warn_class

Warning classes (that may be enabled/disabled).

Enumerator

YASM_WARN_NONE No warning.

YASM_WARN_GENERAL Non-specific warnings.

YASM_WARN_UNREC_CHAR Unrecognized characters (while tokenizing)

YASM_WARN_PREPROC Preprocessor warnings.

YASM_WARN_ORPHAN_LABEL Label alone on a line without a colon.

YASM_WARN_UNINIT_CONTENTS Uninitialized space in code/data section.

YASM_WARN_SIZE_OVERRIDE Double size override.

YASM_WARN_IMPLICIT_SIZE_OVERRIDE Implicit size override.

Definition at line 38 of file errwarn.h.

4.7.5 Function Documentation

4.7.5.1 YASM_LIB_DECL char* yasm__conv_unprint (int *ch*)

Convert a possibly unprintable character into a printable string.

Parameters

ch	possibly unprintable character

Returns

Printable string representation (static buffer).

4.7.5.2 YASM_LIB_DECL void yasm__fatal (const char * *message*, ...)

Reporting point of fatal errors, with variable arguments (internal only).

Warning

This function calls yasm_fatal, and thus does not return to the calling code.

Parameters

message	fatal error message
...	argument list for message

4.7.5.3 YASM_LIB_DECL void yasm_error_clear (void)

Unconditionally clear the error indicator, freeing any associated data.

Has no effect if the error indicator is not set.

4.7.5.4 YASM_LIB_DECL void yasm_error_fetch (yasm_error_class * *eclass*, char ** *str*, unsigned long * *xrefline*, char ** *xrefstr*)

Fetch the error indicator and all associated data.

If the error indicator is set, the output pointers are set to the current error indicator values, and the error indicator is cleared. The code using this function is then responsible for yasm_xfree()'ing str and xrefstr (if non-NULL). If the error indicator is not set, all output values are set to 0 (including eclass, which is set to YASM_ERROR_NONE).

Parameters

eclass	error class (output)
str	error message
xrefline	virtual line used for cross-referencing (0 if no xref)
xrefstr	cross-reference error message (NULL if no xref)

4.7.5.5 YASM_LIB_DECL int yasm_error_matches (yasm_error_class *eclass*)

Check the error indicator against an error class.

To check if any error has been set, check against the YASM_ERROR_GENERAL class. This function properly checks error subclasses.

Parameters

eclass	base error class to check against

Returns

Nonzero if error indicator is set and a subclass of eclass, 0 otherwise.

4.7.5.6 yasm_error_class yasm_error_occurred (void)

Get the error indicator.

YASM_ERROR_NONE is returned if no error has been set. Note that as YASM_ERROR_NONE is 0, the return value can also be treated as a boolean value.

Returns

Current error indicator.

4.7.5.7 YASM_LIB_DECL void yasm_error_set (yasm_error_class *eclass*, const char * *format*, ...)

Set the error indicator.

Has no effect if the error indicator is already set.

Parameters

eclass	error class
format	printf format string
...	argument list for format

4.7.5.8 YASM_LIB_DECL void yasm_error_set_va (yasm_error_class *eclass*, const char * *format*, va_list *va*)

Set the error indicator (va_list version).

Has no effect if the error indicator is already set.

Parameters

eclass	error class
format	printf format string
va	argument list for format

4.7.5.9 YASM_LIB_DECL void yasm_error_set_xref (unsigned long *xrefline*, const char * *format*, ...)

Set a cross-reference for a new error.

Has no effect if the error indicator is already set (e.g. with yasm_error_set()). This function must be called prior to its corresponding yasm_error_set() call.

Parameters

xrefline	virtual line to cross-reference to (should not be 0)
format	printf format string
...	argument list for format

4.7.5.10 YASM_LIB_DECL void yasm_error_set_xref_va (unsigned long *xrefline*, const char * *format*, va_list *va*)

Set a cross-reference for a new error (va_list version).

Has no effect if the error indicator is already set (e.g. with yasm_error_set()). This function must be called prior to its corresponding yasm_error_set() call.

Parameters

xrefline	virtual line to cross-reference to (should not be 0)
format	printf format string
va	argument list for format

4.7.5.11 YASM_LIB_DECL void yasm_errwarn_initialize (void)

Initialize any internal data structures.

4.7.5.12 YASM_LIB_DECL void yasm_errwarn_propagate (yasm_errwarns * *errwarns*, unsigned long *line*)

Propagate error indicator and warning indicator(s) to an error/warning set.

Has no effect if the error indicator and warning indicator are not set. Does not print immediately; yasm_errwarn_output_all() outputs accumulated errors and warnings. Generally multiple errors on the same line will be reported, but errors of class YASM_ERROR_PARSE will get overwritten by any other class on the same line.

Parameters

errwarns	error/warning set
line	virtual line

4.7.5.13 YASM_LIB_DECL yasm_errwarns* yasm_errwarns_create (void)

Create an error/warning set for collection of multiple error/warnings.

Returns

Newly allocated set.

4.7.5.14 YASM_LIB_DECL void yasm_errwarns_destroy (yasm_errwarns * *errwarns*)

Destroy an error/warning set.

Parameters

errwarns	error/warning set

4.7.5.15 YASM_LIB_DECL unsigned int yasm_errwarns_num_errors (yasm_errwarns * *errwarns,* int *warning_as_error*)

Get total number of errors logged.

Parameters

errwarns	error/warning set
warning_as_error	if nonzero, warnings are treated as errors.

Returns

Number of errors.

4.7.5.16 YASM_LIB_DECL void yasm_errwarns_output_all (yasm_errwarns * *errwarns,* yasm_linemap * *lm,* int *warning_as_error,* yasm_print_error_func *print_error,* yasm_print_warning_func *print_warning*)

Outputs error/warning set in sorted order (sorted by virtual line number).

Parameters

errwarns	error/warning set
lm	line map (to convert virtual lines into filename/line pairs)
warning_as_error	if nonzero, treat warnings as errors.
print_error	function called to print out errors
print_warning	function called to print out warnings

4.7.5.17 YASM_LIB_DECL void yasm_warn_clear (void)

Unconditionally clear all warning indicators, freeing any associated data.

Has no effect if no warning indicators have been set.

4.7.5.18 YASM_LIB_DECL void yasm_warn_disable (yasm_warn_class *wclass*)

Disable a class of warnings.

Parameters

wclass	warning class

4.7.5.19 YASM_LIB_DECL void yasm_warn_disable_all (void)

Disable all classes of warnings.

4.7.5.20 YASM_LIB_DECL void yasm_warn_enable (yasm_warn_class *wclass*)

Enable a class of warnings.

Parameters

wclass	warning class

4.7.5.21 YASM_LIB_DECL void yasm_warn_fetch (yasm_warn_class * *wclass*, char ** *str*)

Fetch the first warning indicator and all associated data.

If there is at least one warning indicator, the output pointers are set to the first warning indicator values, and first warning indicator is removed. The code using this function is then responsible for yasm_xfree()'ing str and xrefstr (if non-NULL). If there is no warning indicator set, all output values are set to 0 (including wclass, which is set to YASM_WARN_NONE).

Parameters

wclass	warning class (output)
str	warning message

4.7.5.22 YASM_LIB_DECL yasm_warn_class yasm_warn_occurred (void)

Get the first warning indicator.

YASM_WARN_NONE is returned if no warning has been set. Note that as YASM_WARN_NONE is 0, the return value can also be treated as a boolean value.

Returns

First warning indicator.

4.7.5.23 YASM_LIB_DECL void yasm_warn_set (yasm_warn_class *wclass*, const char * *format*, ...)

Add a warning indicator.

Parameters

wclass	warning class
format	printf format string
...	argument list for format

4.7.5.24 YASM_LIB_DECL void yasm_warn_set_va (yasm_warn_class *wclass,* const char * *format,* va_list *va*)

Add a warning indicator (va_list version).

Parameters

wclass	warning class
format	printf format string
va	argument list for format

4.7.6 Variable Documentation

4.7.6.1 YASM_LIB_DECL void(* yasm_fatal)(const char *message, va_list va)

Reporting point of fatal errors.

Warning

This function must NOT return to calling code; exit or longjmp instead.

Parameters

message	fatal error message
va	va_list argument list for message

4.7.6.2 YASM_LIB_DECL const char*(* yasm_gettext_hook)(const char *msgid)

Hook for library users to map to gettext() if GNU gettext is being used.

Parameters

msgid	message catalog identifier

Returns

Translated message.

4.7.6.3 YASM_LIB_DECL void(* yasm_internal_error_)(const char *file, unsigned int line, const char *message)

Reporting point of internal errors.

These are usually due to sanity check failures in the code.

Warning

This function must NOT return to calling code; exit or longjmp instead.

Parameters

file	source file (ala **FILE**)
line	source line (ala **LINE**)
message	internal error message

4.8 libyasm/expr.h File Reference

YASM expression interface.

Data Structures

- struct yasm_expr__item

  Expression item.
- struct yasm_expr

  Expression.

Macros

- #define **YASM_LIB_DECL**
- #define yasm_expr_create_tree(l, o, r, i) yasm_expr_create ((o), yasm_expr_expr(l), yasm_expr_expr(r), i)

  Create a new expression tree e=l op r.
- #define yasm_expr_create_branch(o, r, i) yasm_expr_create ((o), yasm_expr_expr(r), (yasm_expr__item *)NULL, i)

  Create a new expression branch e=op r.
- #define yasm_expr_create_ident(r, i) yasm_expr_create (YASM_EXPR_IDENT, (r), (yasm_expr__item *)NULL, i)

  Create a new expression identity e=r.
- #define yasm_expr_simplify(e, cbd) yasm_expr__level_tree(e, 1, 1, 1, cbd, NULL, NULL)

  Simplify an expression as much as possible.

Typedefs

- typedef enum yasm_expr__type yasm_expr__type

  Type of an expression item.
- typedef struct yasm_expr__item yasm_expr__item

  Expression item.
- typedef yasm_expr *(* yasm_expr_xform_func)(yasm_expr *e, void *d)

  Extra transformation function for yasm_expr__level_tree().

Enumerations

- enum yasm_expr__type {
  YASM_EXPR_NONE = 0, YASM_EXPR_REG = 1<<0, YASM_EXPR_INT = 1<<1, YASM_EXPR_SUBST = 1<<2,
  YASM_EXPR_FLOAT = 1<<3, YASM_EXPR_SYM = 1<<4, YASM_EXPR_PRECBC = 1<<5, YASM_EXPR-_EXPR = 1<<6 }

  Type of an expression item.

Functions

- YASM_LIB_DECL yasm_expr * yasm_expr_create (yasm_expr_op op, yasm_expr__item *a, yasm_expr__item *b, unsigned long line)

  Create a new expression e=a op b.
- YASM_LIB_DECL yasm_expr__item * yasm_expr_precbc (yasm_bytecode *precbc)

  Create a new preceding-bytecode expression item.
- YASM_LIB_DECL yasm_expr__item * yasm_expr_sym (yasm_symrec *sym)

  Create a new symbol expression item.
- YASM_LIB_DECL yasm_expr__item * yasm_expr_expr (yasm_expr *e)

  Create a new expression expression item.
- YASM_LIB_DECL yasm_expr__item * yasm_expr_int (yasm_intnum *intn)

  Create a new intnum expression item.
- YASM_LIB_DECL yasm_expr__item * yasm_expr_float (yasm_floatnum *flt)

  Create a new floatnum expression item.
- YASM_LIB_DECL yasm_expr__item * yasm_expr_reg (uintptr_t reg)

  Create a new register expression item.
- yasm_expr * yasm_expr_copy (const yasm_expr *e)

  Duplicate an expression.
- YASM_LIB_DECL void yasm_expr_destroy (yasm_expr *e)

  Destroy (free allocated memory for) an expression.
- YASM_LIB_DECL int yasm_expr_is_op (const yasm_expr *e, yasm_expr_op op)

  Determine if an expression is a specified operation (at the top level).
- YASM_LIB_DECL yasm_expr * yasm_expr__level_tree (yasm_expr *e, int fold_const, int simplify_ident, int simplify_reg_mul, int calc_bc_dist, yasm_expr_xform_func expr_xform_extra, void *expr_xform_extra_data)

  Level an entire expression tree.
- YASM_LIB_DECL yasm_expr * yasm_expr_extract_deep_segoff (yasm_expr **ep)

  Extract the segment portion of an expression containing SEG:OFF, leaving the offset.
- YASM_LIB_DECL yasm_expr * yasm_expr_extract_segoff (yasm_expr **ep)

  Extract the segment portion of a SEG:OFF expression, leaving the offset.
- YASM_LIB_DECL yasm_expr * yasm_expr_extract_wrt (yasm_expr **ep)

  Extract the right portion (y) of a x WRT y expression, leaving the left portion (x).
- YASM_LIB_DECL yasm_intnum * yasm_expr_get_intnum (yasm_expr **ep, int calc_bc_dist)

  Get the integer value of an expression if it's just an integer.
- YASM_LIB_DECL const yasm_symrec * yasm_expr_get_symrec (yasm_expr **ep, int simplify)

  Get the symbol value of an expression if it's just a symbol.
- YASM_LIB_DECL const uintptr_t * yasm_expr_get_reg (yasm_expr **ep, int simplify)

  Get the register value of an expression if it's just a register.
- YASM_LIB_DECL void yasm_expr_print (const yasm_expr *e, FILE *f)

  Print an expression.
- YASM_LIB_DECL unsigned int yasm_expr_size (const yasm_expr *e)

  Return the size of an expression, if the user provided it.
- YASM_LIB_DECL const char * yasm_expr_segment (const yasm_expr *e)

  Return the segment of an expression, if the user provided it.
- YASM_LIB_DECL int yasm_expr__traverse_leaves_in_const (const yasm_expr *e, void *d, int(*func)(const yasm_expr__item *ei, void *d))

  Traverse over expression tree in order (const version).

- YASM_LIB_DECL int yasm_expr__traverse_leaves_in (yasm_expr *e, void *d, int(*func)(yasm_expr__item *ei, void *d))

  Traverse over expression tree in order.
- YASM_LIB_DECL void yasm_expr__order_terms (yasm_expr *e)

  Reorder terms of e into canonical order.
- YASM_LIB_DECL yasm_expr * yasm_expr__copy_except (const yasm_expr *e, int except)

  Copy entire expression EXCEPT for index "except" at top level only.
- YASM_LIB_DECL int yasm_expr__contains (const yasm_expr *e, yasm_expr__type t)

  Test if expression contains an item.
- YASM_LIB_DECL int yasm_expr__bc_dist_subst (yasm_expr **ep, void *cbd, void(*callback)(unsigned int subst, yasm_bytecode *precbc, yasm_bytecode *precbc2, void *cbd))

  Transform symrec-symrec terms in expression into YASM_EXPR_SUBST items.
- YASM_LIB_DECL int yasm_expr__subst (yasm_expr *e, unsigned int num_items, const yasm_expr__item *items)

  Substitute items into expr YASM_EXPR_SUBST items (by index).

4.8.1 Detailed Description

YASM expression interface.

Definition in file expr.h.

4.8.2 Macro Definition Documentation

4.8.2.1 #define yasm_expr_create_branch(*o, r, i*) yasm_expr_create ((o), yasm_expr_expr(r), (yasm_expr__item *)NULL, i)

Create a new expression branch e=op r.

Parameters

o	operation
r	expression for right side of new expression
i	line index

Returns

Newly allocated expression.

Definition at line 152 of file expr.h.

4.8.2.2 #define yasm_expr_create_ident(*r, i*) yasm_expr_create (YASM_EXPR_IDENT, (r), (yasm_expr__item *)NULL, i)

Create a new expression identity e=r.

Parameters

r	expression for identity within new expression
i	line index

Returns

Newly allocated expression.

Definition at line 160 of file expr.h.

4.8.2.3 #define yasm_expr_create_tree(*l, o, r, i*) yasm_expr_create ((o), yasm_expr_expr(l), yasm_expr_expr(r), i)

Create a new expression tree e=l op r.

Parameters

l	expression for left side of new expression
o	operation
r	expression for right side of new expression
i	line index

Returns

Newly allocated expression.

Definition at line 143 of file expr.h.

4.8.2.4 #define yasm_expr_simplify(*e, cbd*) yasm_expr__level_tree(e, 1, 1, 1, cbd, NULL, NULL)

Simplify an expression as much as possible.

Eliminates extraneous branches and simplifies integer-only subexpressions. Simplified version of yasm_expr__level_tree().

Parameters

e	expression
cbd	if distance between bytecodes should be calculated

Returns

Simplified expression.

Definition at line 222 of file expr.h.

4.8.3 Typedef Documentation

4.8.3.1 typedef struct yasm_expr__item yasm_expr__item

Expression item.

4.8.3.2 typedef enum yasm_expr__type yasm_expr__type

Type of an expression item.

Types are listed in canonical sorting order. See expr_order_terms(). Note YASM_EXPR_PRECBC must be used carefully (in a-b pairs), as only symrecs can become the relative term in a yasm_value.

4.8.3.3 typedef yasm_expr*(* yasm_expr_xform_func)(yasm_expr *e,void *d)

Extra transformation function for yasm_expr__level_tree().

Parameters

e	expression being simplified
d	data provided as expr_xform_extra_data to yasm_expr__level_tree()

Returns

Transformed e.

Definition at line 194 of file expr.h.

4.8.4 Enumeration Type Documentation

4.8.4.1 enum yasm_expr__type

Type of an expression item.

Types are listed in canonical sorting order. See expr_order_terms(). Note YASM_EXPR_PRECBC must be used carefully (in a-b pairs), as only symrecs can become the relative term in a yasm_value.

Enumerator

YASM_EXPR_NONE Nothing.

YASM_EXPR_REG Register.

YASM_EXPR_INT Integer value.

YASM_EXPR_SUBST Substitution placeholder.

YASM_EXPR_FLOAT Floating point value.

YASM_EXPR_SYM Symbol.

YASM_EXPR_PRECBC Direct bytecode ref (rather than via sym)

YASM_EXPR_EXPR Subexpression.

Definition at line 42 of file expr.h.

4.8.5 Function Documentation

4.8.5.1 YASM_LIB_DECL int yasm_expr__bc_dist_subst (yasm_expr ** *ep*, void * *cbd*, void(*)(unsigned int subst, yasm_bytecode *precbc, yasm_bytecode *precbc2, void *cbd) *callback*)

Transform symrec-symrec terms in expression into YASM_EXPR_SUBST items.

Calls the callback function for each symrec-symrec term.

Parameters

ep	expression (pointer to)
cbd	callback data passed to callback function
callback	callback function: given subst index for bytecode pair, bytecode pair (bc2-bc1), and cbd (callback data)

Returns

Number of transformations made.

4.8.5.2 YASM_LIB_DECL int yasm_expr__contains (const yasm_expr * e, yasm_expr__type t)

Test if expression contains an item.

Searches recursively into subexpressions.

Parameters

e	expression
t	type of item to look for

Returns

Nonzero if expression contains an item of type t, zero if not.

4.8.5.3 YASM_LIB_DECL yasm_expr* yasm_expr__copy_except (const yasm_expr * e, int except)

Copy entire expression EXCEPT for index "except" at *top level only*.

Parameters

e	expression
except	term index not to copy; -1 to copy all terms

Returns

Newly allocated copy of expression.

4.8.5.4 YASM_LIB_DECL yasm_expr* yasm_expr__level_tree (yasm_expr * e, int fold_const, int simplify_ident, int simplify_reg_mul, int calc_bc_dist, yasm_expr_xform_func expr_xform_extra, void * expr_xform_extra_data)

Level an entire expression tree.

Parameters

e	expression
fold_const	enable constant folding if nonzero
simplify_ident	simplify identities
simplify_reg_mul	simplify REG*1 identities
calc_bc_dist	nonzero if distances between bytecodes should be calculated, 0 if they should be left intact
expr_xform_extra	extra transformation function
expr_xform_extra_data	data to pass to expr_xform_extra

Returns

Leveled expression.

4.8.5.5 YASM_LIB_DECL void yasm_expr__order_terms (yasm_expr * e)

Reorder terms of e into canonical order.

Only reorders if reordering doesn't change meaning of expression. (eg, doesn't reorder SUB). Canonical order: REG, INT, FLOAT, SYM, EXPR. Multiple terms of a single type are kept in the same order as in the original expression.

Parameters

e	expression

Note

Only performs reordering on *one* level (no recursion).

4.8.5.6 YASM_LIB_DECL int yasm_expr__subst (yasm_expr * e, unsigned int num_items, const yasm_expr__item * items)

Substitute items into expr YASM_EXPR_SUBST items (by index).

Items are copied, so caller is responsible for freeing array of items.

Parameters

e	expression
num_items	number of items in items array
items	items array

Returns

1 on error (index out of range).

4.8.5.7 YASM_LIB_DECL int yasm_expr__traverse_leaves_in (yasm_expr * e, void * d, int(*)(yasm_expr__item *ei, void *d) func)

Traverse over expression tree in order.

Calls func for each leaf (non-operation).

Parameters

e	expression
d	data passed to each call to func
func	callback function

Returns

Stops early (and returns 1) if func returns 1. Otherwise returns 0.

4.8.5.8 YASM_LIB_DECL int yasm_expr__traverse_leaves_in_const (const yasm_expr * e, void * d, int(*)(const yasm_expr__item *ei, void *d) func)

Traverse over expression tree in order (const version).

Calls func for each leaf (non-operation).

Parameters

e	expression
d	data passed to each call to func
func	callback function

Returns

Stops early (and returns 1) if func returns 1. Otherwise returns 0.

4.8.5.9 yasm_expr* yasm_expr_copy (const yasm_expr * e)

Duplicate an expression.

Parameters

e	expression

Returns

Newly allocated expression identical to e.

4.8.5.10 YASM_LIB_DECL yasm_expr* yasm_expr_create (yasm_expr_op *op*, yasm_expr__item * *a*, yasm_expr__item * *b*, unsigned long *line*)

Create a new expression e=a op b.

Parameters

op	operation
a	expression item a
b	expression item b (optional depending on op)
line	virtual line (where expression defined)

Returns

Newly allocated expression.

4.8.5.11 YASM_LIB_DECL void yasm_expr_destroy (yasm_expr * e)

Destroy (free allocated memory for) an expression.

Parameters

e	expression

4.8.5.12 YASM_LIB_DECL yasm_expr__item* yasm_expr_expr (yasm_expr * e)

Create a new expression expression item.

Parameters

e	expression

Returns

Newly allocated expression item.

4.8.5.13 YASM_LIB_DECL yasm_expr* yasm_expr_extract_deep_segoff (yasm_expr ** *ep*)

Extract the segment portion of an expression containing SEG:OFF, leaving the offset.

Parameters

ep	expression (pointer to)

Returns

NULL if unable to extract a segment (expr does not contain a YASM_EXPR_SEGOFF operator), otherwise the segment expression. The input expression is modified such that on return, it's the offset expression.

4.8.5.14 YASM_LIB_DECL yasm_expr* yasm_expr_extract_segoff (yasm_expr ** *ep*)

Extract the segment portion of a SEG:OFF expression, leaving the offset.

Parameters

ep	expression (pointer to)

Returns

NULL if unable to extract a segment (YASM_EXPR_SEGOFF not the top-level operator), otherwise the segment expression. The input expression is modified such that on return, it's the offset expression.

4.8.5.15 YASM_LIB_DECL yasm_expr* yasm_expr_extract_wrt (yasm_expr ** *ep*)

Extract the right portion (y) of a x WRT y expression, leaving the left portion (x).

Parameters

ep	expression (pointer to)

Returns

NULL if unable to extract (YASM_EXPR_WRT not the top-level operator), otherwise the right side of the WRT expression. The input expression is modified such that on return, it's the left side of the WRT expression.

4.8.5.16 YASM_LIB_DECL yasm_expr__item* yasm_expr_float (yasm_floatnum * *flt*)

Create a new floatnum expression item.

Parameters

flt	floatnum

Returns

Newly allocated expression item.

4.8.5.17 YASM_LIB_DECL yasm_intnum* yasm_expr_get_intnum (yasm_expr ** *ep,* int *calc_bc_dist*)

Get the integer value of an expression if it's just an integer.

Parameters

ep	expression (pointer to)
calc_bc_dist	nonzero if distances between bytecodes should be calculated, 0 if NULL should be returned in this case

Returns

NULL if the expression is too complex (contains anything other than integers, ie floats, non-valued labels, registers); otherwise the intnum value of the expression.

4.8.5.18 YASM_LIB_DECL const uintptr_t* yasm_expr_get_reg (yasm_expr ** *ep,* int *simplify*)

Get the register value of an expression if it's just a register.

Parameters

ep	expression (pointer to)
simplify	if nonzero, simplify the expression first

Returns

NULL if the expression is too complex; otherwise the register value of the expression.

4.8.5.19 YASM_LIB_DECL const yasm_symrec* yasm_expr_get_symrec (yasm_expr ** *ep,* int *simplify*)

Get the symbol value of an expression if it's just a symbol.

Parameters

ep	expression (pointer to)
simplify	if nonzero, simplify the expression first

Returns

NULL if the expression is too complex; otherwise the symbol value of the expression.

4.8.5.20 YASM_LIB_DECL yasm_expr__item* yasm_expr_int (yasm_intnum * *intn*)

Create a new intnum expression item.

Parameters

intn	intnum

Returns

Newly allocated expression item.

4.8.5.21 YASM_LIB_DECL int yasm_expr_is_op (const yasm_expr * e, yasm_expr_op op)

Determine if an expression is a specified operation (at the top level).

Parameters

e	expression
op	operator

Returns

Nonzero if the expression was the specified operation at the top level, zero otherwise.

4.8.5.22 YASM_LIB_DECL yasm_expr__item* yasm_expr_precbc (yasm_bytecode * precbc)

Create a new preceding-bytecode expression item.

Parameters

precbc	preceding bytecode

Returns

Newly allocated expression item.

4.8.5.23 YASM_LIB_DECL void yasm_expr_print (const yasm_expr * e, FILE * f)

Print an expression.

For debugging purposes.

Parameters

e	expression
f	file

4.8.5.24 YASM_LIB_DECL yasm_expr__item* yasm_expr_reg (uintptr_t reg)

Create a new register expression item.

Parameters

reg	register

Returns

Newly allocated expression item.

4.8.5.25 YASM_LIB_DECL const char* yasm_expr_segment (const yasm_expr * e)

Return the segment of an expression, if the user provided it.

Parameters

e	expression

4.8.5.26 YASM_LIB_DECL unsigned int yasm_expr_size (const yasm_expr * e)

Return the size of an expression, if the user provided it.

Parameters

e	expression

4.8.5.27 YASM_LIB_DECL yasm_expr__item* yasm_expr_sym (yasm_symrec * sym)

Create a new symbol expression item.

Parameters

sym	symbol

Returns

Newly allocated expression item.

4.9 libyasm/file.h File Reference

YASM file helpers.

Data Structures

- struct yasm_scanner

  Re2c scanner state.

Macros

- #define **YASM_LIB_DECL**
- #define yasm__splitpath(path, tail) yasm__splitpath_unix(path, tail)

  Split a pathname into head (directory) and tail (base filename) portions.
- #define yasm__combpath(from, to) yasm__combpath_unix(from, to)

  Build a pathname that is equivalent to accessing the "to" pathname when you're in the directory containing "from".
- #define yasm__createpath(path) yasm__createpath_common(path, 0)

  Recursively create tree of directories needed for pathname.
- #define YASM_WRITE_8(ptr, val) *((ptr)++) = (unsigned char)((val) & 0xFF)

  Write an 8-bit value to a buffer, incrementing buffer pointer.
- #define YASM_WRITE_16_L(ptr, val)

  Write a 16-bit value to a buffer in little endian, incrementing buffer pointer.
- #define YASM_WRITE_32_L(ptr, val)

  Write a 32-bit value to a buffer in little endian, incrementing buffer pointer.
- #define YASM_WRITE_16_B(ptr, val)

  Write a 16-bit value to a buffer in big endian, incrementing buffer pointer.

- #define YASM_WRITE_32_B(ptr, val)

  Write a 32-bit value to a buffer in big endian, incrementing buffer pointer.
- #define YASM_SAVE_8(ptr, val) *(ptr) = (unsigned char)((val) & 0xFF)

  Write an 8-bit value to a buffer.
- #define YASM_SAVE_16_L(ptr, val)

  Write a 16-bit value to a buffer in little endian.
- #define YASM_SAVE_32_L(ptr, val)

  Write a 32-bit value to a buffer in little endian.
- #define YASM_SAVE_16_B(ptr, val)

  Write a 16-bit value to a buffer in big endian.
- #define YASM_SAVE_32_B(ptr, val)

  Write a 32-bit value to a buffer in big endian.
- #define YASM_READ_8(val, ptr) (val) = *((ptr)++) & 0xFF

  Read an 8-bit value from a buffer, incrementing buffer pointer.
- #define YASM_READ_16_L(val, ptr)

  Read a 16-bit value from a buffer in little endian, incrementing buffer pointer.
- #define YASM_READ_32_L(val, ptr)

  Read a 32-bit value from a buffer in little endian, incrementing buffer pointer.
- #define YASM_READ_16_B(val, ptr)

  Read a 16-bit value from a buffer in big endian, incrementing buffer pointer.
- #define YASM_READ_32_B(val, ptr)

  Read a 32-bit value from a buffer in big endian, incrementing buffer pointer.
- #define YASM_LOAD_8(val, ptr) (val) = *(ptr) & 0xFF

  Read an 8-bit value from a buffer.
- #define YASM_LOAD_16_L(val, ptr)

  Read a 16-bit value from a buffer in little endian.
- #define YASM_LOAD_32_L(val, ptr)

  Read a 32-bit value from a buffer in little endian.
- #define YASM_LOAD_16_B(val, ptr)

  Read a 16-bit value from a buffer in big endian.
- #define YASM_LOAD_32_B(val, ptr)

  Read a 32-bit value from a buffer in big endian.

Typedefs

- typedef struct yasm_scanner yasm_scanner

  Re2c scanner state.

Functions

- YASM_LIB_DECL void yasm_scanner_initialize (yasm_scanner *scanner)

  Initialize scanner state.
- YASM_LIB_DECL void yasm_scanner_delete (yasm_scanner *scanner)

  Frees any memory used by scanner state; does not free state itself.
- YASM_LIB_DECL int yasm_fill_helper (yasm_scanner *scanner, unsigned char **cursor, size_t(*input_func)(void *d, unsigned char *buf, size_t max), void *input_func_data)

  Fill a scanner state structure with data coming from an input function.

- YASM_LIB_DECL void yasm_unescape_cstring (unsigned char *str, size_t *len)

  Unescape a string with C-style escapes.
- YASM_LIB_DECL size_t yasm__splitpath_unix (const char *path, const char **tail)

  Split a UNIX pathname into head (directory) and tail (base filename) portions.
- YASM_LIB_DECL size_t yasm__splitpath_win (const char *path, const char **tail)

  Split a Windows pathname into head (directory) and tail (base filename) portions.
- YASM_LIB_DECL char * yasm__getcwd (void)

  Get the current working directory.
- YASM_LIB_DECL char * yasm__abspath (const char *path)

  Convert a relative or absolute pathname into an absolute pathname.
- YASM_LIB_DECL char * yasm__combpath_unix (const char *from, const char *to)

  Build a UNIX pathname that is equivalent to accessing the "to" pathname when you're in the directory containing "from".
- YASM_LIB_DECL char * yasm__combpath_win (const char *from, const char *to)

  Build a Windows pathname that is equivalent to accessing the "to" pathname when you're in the directory containing "from".
- YASM_LIB_DECL size_t yasm__createpath_common (const char *path, int win)

  Recursively create tree of directories needed for pathname.
- YASM_LIB_DECL FILE * yasm_fopen_include (const char *iname, const char *from, const char *mode, char **oname)

  Try to find and open an include file, searching through include paths.
- YASM_LIB_DECL void yasm_delete_include_paths (void)

  Delete any stored include paths added by yasm_add_include_path().
- YASM_LIB_DECL const char * yasm_get_include_dir (void **iter)

  Iterate through include paths.
- YASM_LIB_DECL void yasm_add_include_path (const char *path)

  Add an include path for use by yasm_fopen_include().
- YASM_LIB_DECL size_t yasm_fwrite_16_l (unsigned short val, FILE *f)

  Direct-to-file version of YASM_SAVE_16_L().
- YASM_LIB_DECL size_t yasm_fwrite_32_l (unsigned long val, FILE *f)

  Direct-to-file version of YASM_SAVE_32_L().
- YASM_LIB_DECL size_t yasm_fwrite_16_b (unsigned short val, FILE *f)

  Direct-to-file version of YASM_SAVE_16_B().
- YASM_LIB_DECL size_t yasm_fwrite_32_b (unsigned long val, FILE *f)

  Direct-to-file version of YASM_SAVE_32_B().

4.9.1 Detailed Description

YASM file helpers.

Definition in file file.h.

4.9.2 Macro Definition Documentation

4.9.2.1 #define yasm__combpath(*from, to*) yasm__combpath_unix(from, to)

Build a pathname that is equivalent to accessing the "to" pathname when you're in the directory containing "from".

Result is relative if both from and to are relative. Unless otherwise defined, defaults to yasm__combpath_unix().

Parameters

from	from pathname
to	to pathname

Returns

Combined path (newly allocated).

Definition at line 174 of file file.h.

4.9.2.2 #define yasm__createpath(*path*) yasm__createpath_common(path, 0)

Recursively create tree of directories needed for pathname.

Unless otherwise defined, defaults to yasm__createpath_unix().

Parameters

path	pathname

Returns

Length of directory portion of pathname.

Definition at line 198 of file file.h.

4.9.2.3 #define yasm__splitpath(*path, tail*) yasm__splitpath_unix(path, tail)

Split a pathname into head (directory) and tail (base filename) portions.

Unless otherwise defined, defaults to yasm__splitpath_unix().

Parameters

path	pathname
tail	(returned) base filename

Returns

Length of head (directory).

Definition at line 119 of file file.h.

4.9.2.4 #define YASM_LOAD_16_B(*val, ptr*)

Value:

```
do {                                    \
            (val) = (*(ptr) & 0xFF) << 8;       \
            (val) |= *((ptr)+1) & 0xFF;         \
        } while (0)
```

Read a 16-bit value from a buffer in big endian.

Does not increment buffer pointer.

Note

Only works properly if ptr is an (unsigned char *).

Parameters

ptr	buffer
val	16-bit value

Definition at line 505 of file file.h.

4.9.2.5 #define YASM_LOAD_16_L(*val*, *ptr*)

Value:

```
do {                                    \
            (val) = *(ptr) & 0xFF;              \
            (val) |= (*((ptr)+1) & 0xFF) << 8;  \
        } while (0)
```

Read a 16-bit value from a buffer in little endian.

Does not increment buffer pointer.

Note

Only works properly if ptr is an (unsigned char *).

Parameters

ptr	buffer
val	16-bit value

Definition at line 479 of file file.h.

4.9.2.6 #define YASM_LOAD_32_B(*val*, *ptr*)

Value:

```
do {                                    \
            (val) = (unsigned long)((*(ptr) & 0xFF) << 24);         \
            (val) |= (unsigned long)((*((ptr)+1) & 0xFF) << 16);    \
            (val) |= (unsigned long)((*((ptr)+2) & 0xFF) << 8);     \
            (val) |= (unsigned long)(*((ptr)+3) & 0xFF);            \
        } while (0)
```

Read a 32-bit value from a buffer in big endian.

Does not increment buffer pointer.

Note

Only works properly if ptr is an (unsigned char *).

Parameters

ptr	buffer
val	32-bit value

Definition at line 517 of file file.h.

4.9.2.7 #define YASM_LOAD_32_L(*val*, *ptr*)

Value:

```
do {                                    \
            (val) = (unsigned long)(*(ptr) & 0xFF);                 \
            (val) |= (unsigned long)((*((ptr)+1) & 0xFF) << 8);     \
            (val) |= (unsigned long)((*((ptr)+2) & 0xFF) << 16);    \
            (val) |= (unsigned long)((*((ptr)+3) & 0xFF) << 24);    \
        } while (0)
```

Read a 32-bit value from a buffer in little endian.

Does not increment buffer pointer.

Note

Only works properly if ptr is an (unsigned char *).

Parameters

ptr	buffer
val	32-bit value

Definition at line 491 of file file.h.

4.9.2.8 #define YASM_LOAD_8(*val*, *ptr*) (val) = *(ptr) & 0xFF

Read an 8-bit value from a buffer.

Does not increment buffer pointer.

Note

Only works properly if ptr is an (unsigned char *).

Parameters

ptr	buffer
val	8-bit value

Definition at line 470 of file file.h.

4.9.2.9 #define YASM_READ_16_B(*val*, *ptr*)

Value:

```
do {                                    \
            (val) = (*((ptr)++) & 0xFF) << 8;   \
            (val) |= *((ptr)++) & 0xFF;         \
        } while (0)
```

Read a 16-bit value from a buffer in big endian, incrementing buffer pointer.

Note

Only works properly if ptr is an (unsigned char *).

Parameters

ptr	buffer
val	16-bit value

Definition at line 445 of file file.h.

4.9.2.10 #define YASM_READ_16_L(*val, ptr*)

Value:

```
do {                                    \
            (val) = *((ptr)++) & 0xFF;          \
            (val) |= (*((ptr)++) & 0xFF) << 8;  \
        } while (0)
```

Read a 16-bit value from a buffer in little endian, incrementing buffer pointer.

Note

Only works properly if ptr is an (unsigned char *).

Parameters

ptr	buffer
val	16-bit value

Definition at line 419 of file file.h.

4.9.2.11 #define YASM_READ_32_B(*val, ptr*)

Value:

```
do {                                    \
            (val) = (*((ptr)++) & 0xFF) << 24;  \
            (val) |= (*((ptr)++) & 0xFF) << 16; \
            (val) |= (*((ptr)++) & 0xFF) << 8;  \
            (val) |= *((ptr)++) & 0xFF;         \
        } while (0)
```

Read a 32-bit value from a buffer in big endian, incrementing buffer pointer.

Note

Only works properly if ptr is an (unsigned char *).

Parameters

ptr	buffer
val	32-bit value

Definition at line 457 of file file.h.

4.9.2.12 #define YASM_READ_32_L(*val, ptr*)

Value:

```
do {                                    \
            (val) = *((ptr)++) & 0xFF;          \
            (val) |= (*((ptr)++) & 0xFF) << 8;  \
            (val) |= (*((ptr)++) & 0xFF) << 16; \
            (val) |= (*((ptr)++) & 0xFF) << 24; \
        } while (0)
```

Read a 32-bit value from a buffer in little endian, incrementing buffer pointer.

Note

Only works properly if ptr is an (unsigned char *).

Parameters

ptr	buffer
val	32-bit value

Definition at line 431 of file file.h.

4.9.2.13 #define YASM_READ_8(*val, ptr*) (val) = *((ptr)++) & 0xFF

Read an 8-bit value from a buffer, incrementing buffer pointer.

Note

Only works properly if ptr is an (unsigned char *).

Parameters

ptr	buffer
val	8-bit value

Definition at line 410 of file file.h.

4.9.2.14 #define YASM_SAVE_16_B(*ptr, val*)

Value:

```
do {                                    \
            *(ptr) = (unsigned char)(((val) >> 8) & 0xFF);      \
            *((ptr)+1) = (unsigned char)((val) & 0xFF);         \
        } while (0)
```

Write a 16-bit value to a buffer in big endian.

Does not increment buffer pointer.

Note

Only works properly if ptr is an (unsigned char *).

Parameters

ptr	buffer
val	16-bit value

Definition at line 345 of file file.h.

4.9.2.15 #define YASM_SAVE_16_L(*ptr*, *val*)

Value:

```
do {                                    \
            *(ptr) = (unsigned char)((val) & 0xFF);             \
            *((ptr)+1) = (unsigned char)(((val) >> 8) & 0xFF);  \
        } while (0)
```

Write a 16-bit value to a buffer in little endian.

Does not increment buffer pointer.

Note

Only works properly if ptr is an (unsigned char *).

Parameters

ptr	buffer
val	16-bit value

Definition at line 319 of file file.h.

4.9.2.16 #define YASM_SAVE_32_B(*ptr*, *val*)

Value:

```
do {                                    \
            *(ptr) = (unsigned char)(((val) >> 24) & 0xFF);     \
            *((ptr)+1) = (unsigned char)(((val) >> 16) & 0xFF); \
            *((ptr)+2) = (unsigned char)(((val) >> 8) & 0xFF);  \
            *((ptr)+3) = (unsigned char)((val) & 0xFF);         \
        } while (0)
```

Write a 32-bit value to a buffer in big endian.

Does not increment buffer pointer.

Note

Only works properly if ptr is an (unsigned char *).

Parameters

ptr	buffer
val	32-bit value

Definition at line 357 of file file.h.

4.9.2.17 #define YASM_SAVE_32_L(*ptr, val*)

Value:

```
do {                                    \
            *(ptr) = (unsigned char)((val) & 0xFF);             \
            *((ptr)+1) = (unsigned char)(((val) >> 8) & 0xFF);  \
            *((ptr)+2) = (unsigned char)(((val) >> 16) & 0xFF); \
            *((ptr)+3) = (unsigned char)(((val) >> 24) & 0xFF); \
        } while (0)
```

Write a 32-bit value to a buffer in little endian.

Does not increment buffer pointer.

Note

Only works properly if ptr is an (unsigned char *).

Parameters

ptr	buffer
val	32-bit value

Definition at line 331 of file file.h.

4.9.2.18 #define YASM_SAVE_8(*ptr, val*) *(ptr) = (unsigned char)((val) & 0xFF)

Write an 8-bit value to a buffer.

Does not increment buffer pointer.

Note

Only works properly if ptr is an (unsigned char *).

Parameters

ptr	buffer
val	8-bit value

Definition at line 310 of file file.h.

4.9.2.19 #define YASM_WRITE_16_B(*ptr, val*)

Value:

```
do {                                    \
            *((ptr)++) = (unsigned char)(((val) >> 8) & 0xFF);  \
            *((ptr)++) = (unsigned char)((val) & 0xFF);         \
        } while (0)
```

Write a 16-bit value to a buffer in big endian, incrementing buffer pointer.

Note

Only works properly if ptr is an (unsigned char *).

Parameters

ptr	buffer
val	16-bit value

Definition at line 284 of file file.h.

4.9.2.20 #define YASM_WRITE_16_L(*ptr, val*)

Value:

```
do {                                    \
            *((ptr)++) = (unsigned char)((val) & 0xFF);         \
            *((ptr)++) = (unsigned char)(((val) >> 8) & 0xFF);  \
        } while (0)
```

Write a 16-bit value to a buffer in little endian, incrementing buffer pointer.

Note

Only works properly if ptr is an (unsigned char *).

Parameters

ptr	buffer
val	16-bit value

Definition at line 258 of file file.h.

4.9.2.21 #define YASM_WRITE_32_B(*ptr, val*)

Value:

```
do {                                    \
            *((ptr)++) = (unsigned char)(((val) >> 24) & 0xFF); \
            *((ptr)++) = (unsigned char)(((val) >> 16) & 0xFF); \
            *((ptr)++) = (unsigned char)(((val) >> 8) & 0xFF);  \
            *((ptr)++) = (unsigned char)((val) & 0xFF);         \
        } while (0)
```

Write a 32-bit value to a buffer in big endian, incrementing buffer pointer.

Note

Only works properly if ptr is an (unsigned char *).

Parameters

ptr	buffer
val	32-bit value

Definition at line 296 of file file.h.

4.9.2.22 #define YASM_WRITE_32_L(*ptr, val*)

Value:

```
do {                                    \
```

```
    *((ptr)++) = (unsigned char)((val) & 0xFF);         \
    *((ptr)++) = (unsigned char)(((val) >> 8) & 0xFF);  \
    *((ptr)++) = (unsigned char)(((val) >> 16) & 0xFF); \
    *((ptr)++) = (unsigned char)(((val) >> 24) & 0xFF); \
} while (0)
```

Write a 32-bit value to a buffer in little endian, incrementing buffer pointer.

Note

Only works properly if ptr is an (unsigned char *).

Parameters

ptr	buffer
val	32-bit value

Definition at line 270 of file file.h.

4.9.2.23 #define YASM_WRITE_8(*ptr, val*) *((ptr)++) = (unsigned char)((val) & 0xFF)

Write an 8-bit value to a buffer, incrementing buffer pointer.

Note

Only works properly if ptr is an (unsigned char *).

Parameters

ptr	buffer
val	8-bit value

Definition at line 249 of file file.h.

4.9.3 Typedef Documentation

4.9.3.1 typedef struct yasm_scanner yasm_scanner

Re2c scanner state.

4.9.4 Function Documentation

4.9.4.1 YASM_LIB_DECL char* yasm__abspath (const char * *path*)

Convert a relative or absolute pathname into an absolute pathname.

Parameters

path	pathname

Returns

Absolute version of path (newly allocated).

4.9.4.2 YASM_LIB_DECL char* yasm__combpath_unix (const char * *from*, const char * *to*)

Build a UNIX pathname that is equivalent to accessing the "to" pathname when you're in the directory containing "from".

Result is relative if both from and to are relative.

Parameters

from	from pathname
to	to pathname

Returns

Combined path (newly allocated).

4.9.4.3 YASM_LIB_DECL char* yasm__combpath_win (const char * *from*, const char * *to*)

Build a Windows pathname that is equivalent to accessing the "to" pathname when you're in the directory containing "from".

Result is relative if both from and to are relative.

Parameters

from	from pathname
to	to pathname

Returns

Combined path (newly allocated).

4.9.4.4 YASM_LIB_DECL size_t yasm__createpath_common (const char * *path*, int *win*)

Recursively create tree of directories needed for pathname.

Parameters

path	pathname
win	handle windows paths

Returns

Length of directory portion of pathname.

4.9.4.5 YASM_LIB_DECL char* yasm__getcwd (void)

Get the current working directory.

Returns

Current working directory pathname (newly allocated).

4.9.4.6 YASM_LIB_DECL size_t yasm__splitpath_unix (const char * *path*, const char ** *tail*)

Split a UNIX pathname into head (directory) and tail (base filename) portions.

Parameters

path	pathname
tail	(returned) base filename

Returns

Length of head (directory).

4.9.4.7 YASM_LIB_DECL size_t yasm__splitpath_win (const char * *path*, const char ** *tail*)

Split a Windows pathname into head (directory) and tail (base filename) portions.

Parameters

path	pathname
tail	(returned) base filename

Returns

Length of head (directory).

4.9.4.8 YASM_LIB_DECL void yasm_add_include_path (const char * *path*)

Add an include path for use by yasm_fopen_include().

If path is relative, it is treated by yasm_fopen_include() as relative to the current working directory.

Parameters

path	path to add

4.9.4.9 YASM_LIB_DECL int yasm_fill_helper (yasm_scanner * *scanner*, unsigned char ** *cursor*, size_t(*)(void *d, unsigned char *buf, size_t max) *input_func*, void * *input_func_data*)

Fill a scanner state structure with data coming from an input function.

Parameters

scanner	Re2c scanner state
cursor	Re2c scan cursor
input_func	Input function to read data; takes buffer and maximum number of bytes, returns number of bytes read.
input_func_data	Data to pass as the first parameter to input_func

Returns

1 if this was the first time this function was called on this scanner state, 0 otherwise.

4.9.4.10 YASM_LIB_DECL FILE* yasm_fopen_include (const char * *iname*, const char * *from*, const char * *mode*, char ** *oname*)

Try to find and open an include file, searching through include paths.

First iname is looked for relative to the directory containing "from", then it's looked for relative to each of the include paths.

All pathnames may be either absolute or relative; from, oname, and include paths, if relative, are relative from the current working directory.

First match wins; the full pathname (newly allocated) to the opened file is saved into oname, and the fopen'ed FILE * is returned. If not found, NULL is returned.

Parameters

iname	file to include
from	file doing the including
mode	fopen mode string
oname	full pathname of included file (may be relative). NULL may be passed if this is unwanted.

Returns

fopen'ed include file, or NULL if not found.

4.9.4.11 YASM_LIB_DECL size_t yasm_fwrite_16_b (unsigned short *val*, FILE * *f*)

Direct-to-file version of YASM_SAVE_16_B().

Note

Using the macro multiple times with a single fwrite() call will probably be faster than calling this function many times.

Parameters

val	16-bit value
f	file

Returns

1 if the write was successful, 0 if not (just like fwrite()).

4.9.4.12 YASM_LIB_DECL size_t yasm_fwrite_16_l (unsigned short *val*, FILE * *f*)

Direct-to-file version of YASM_SAVE_16_L().

Note

Using the macro multiple times with a single fwrite() call will probably be faster than calling this function many times.

Parameters

val	16-bit value
f	file

Returns

1 if the write was successful, 0 if not (just like fwrite()).

4.9.4.13 YASM_LIB_DECL size_t yasm_fwrite_32_b (unsigned long *val*, FILE * *f*)

Direct-to-file version of YASM_SAVE_32_B().

Note

Using the macro multiple times with a single fwrite() call will probably be faster than calling this function many times.

Parameters

val	32-bit value
f	file

Returns

1 if the write was successful, 0 if not (just like fwrite()).

4.9.4.14 YASM_LIB_DECL size_t yasm_fwrite_32_l (unsigned long *val*, FILE * *f*)

Direct-to-file version of YASM_SAVE_32_L().

Note

Using the macro multiple times with a single fwrite() call will probably be faster than calling this function many times.

Parameters

val	32-bit value
f	file

Returns

1 if the write was successful, 0 if not (just like fwrite()).

4.9.4.15 YASM_LIB_DECL void yasm_scanner_delete (yasm_scanner * *scanner*)

Frees any memory used by scanner state; does not free state itself.

Parameters

scanner	Re2c scanner state

4.9.4.16 YASM_LIB_DECL void yasm_scanner_initialize (yasm_scanner * *scanner*)

Initialize scanner state.

Parameters

scanner	Re2c scanner state

4.9.4.17 YASM_LIB_DECL void yasm_unescape_cstring (unsigned char * *str*, size_t * *len*)

Unescape a string with C-style escapes.

Handles b, f, n, r, t, and hex and octal escapes. String is updated in-place. Edge cases:

- hex escapes: reads as many hex digits as possible, takes last 2 as value.

- oct escapes: takes up to 3 digits 0-9 and scales appropriately, with warning.

Parameters

str	C-style string (updated in place)
len	length of string (updated with new length)

4.10 libyasm/floatnum.h File Reference

YASM floating point (IEEE) interface.

Functions

- YASM_LIB_DECL void yasm_floatnum_initialize (void)

  Initialize floatnum internal data structures.
- YASM_LIB_DECL void yasm_floatnum_cleanup (void)

  Clean up internal floatnum allocations.
- YASM_LIB_DECL yasm_floatnum * yasm_floatnum_create (const char *str)

  Create a new floatnum from a decimal string.
- YASM_LIB_DECL yasm_floatnum * yasm_floatnum_copy (const yasm_floatnum *flt)

  Duplicate a floatnum.
- YASM_LIB_DECL void yasm_floatnum_destroy (yasm_floatnum *flt)

  Destroy (free allocated memory for) a floatnum.
- YASM_LIB_DECL int yasm_floatnum_calc (yasm_floatnum *acc, yasm_expr_op op, yasm_floatnum *operand)

  Floating point calculation function: acc = acc op operand.
- YASM_LIB_DECL int yasm_floatnum_get_int (const yasm_floatnum *flt, unsigned long *ret_val)

  Convert a floatnum to single-precision and return as 32-bit value.
- YASM_LIB_DECL int yasm_floatnum_get_sized (const yasm_floatnum *flt, unsigned char *ptr, size_t destsize, size_t valsize, size_t shift, int bigendian, int warn)

  Output a yasm_floatnum to buffer in little-endian or big-endian.
- YASM_LIB_DECL int yasm_floatnum_check_size (const yasm_floatnum *flt, size_t size)

  Basic check to see if size is valid for flt conversion (using yasm_floatnum_get_sized()).
- YASM_LIB_DECL void yasm_floatnum_print (const yasm_floatnum *flt, FILE *f)

  Print various representations of a floatnum.

4.10.1 Detailed Description

YASM floating point (IEEE) interface.

Definition in file floatnum.h.

4.10.2 Function Documentation

4.10.2.1 YASM_LIB_DECL int yasm_floatnum_calc (yasm_floatnum * *acc*, yasm_expr_op *op*, yasm_floatnum * *operand*)

Floating point calculation function: acc = acc op operand.

Note

Not all operations in yasm_expr_op may be supported; unsupported operations will result in an error.

Parameters

acc	floatnum accumulator
op	operation
operand	floatnum operand

Returns

Nonzero on error.

4.10.2.2 YASM_LIB_DECL int yasm_floatnum_check_size (const yasm_floatnum * *flt*, size_t *size*)

Basic check to see if size is valid for flt conversion (using yasm_floatnum_get_sized()).

Doesn't actually check for underflow/overflow but rather checks for size=32,64,80 (at present).

Parameters

flt	floatnum
size	number of bits of output space

Returns

1 if valid size, 0 if invalid size.

4.10.2.3 YASM_LIB_DECL void yasm_floatnum_cleanup (void)

Clean up internal floatnum allocations.

4.10.2.4 YASM_LIB_DECL yasm_floatnum* yasm_floatnum_copy (const yasm_floatnum * *flt*)

Duplicate a floatnum.

Parameters

flt	floatnum

Returns

Newly allocated floatnum with the same value as flt.

4.10.2.5 YASM_LIB_DECL yasm_floatnum* yasm_floatnum_create (const char * *str*)

Create a new floatnum from a decimal string.

The input string must be in standard C representation ([+-]123.456e[-+]789).

Parameters

str	floating point decimal string

Returns

Newly allocated floatnum.

4.10.2.6 YASM_LIB_DECL void yasm_floatnum_destroy (yasm_floatnum * *flt*)

Destroy (free allocated memory for) a floatnum.

Parameters

flt	floatnum

4.10.2.7 YASM_LIB_DECL int yasm_floatnum_get_int (const yasm_floatnum * *flt,* unsigned long * *ret_val*)

Convert a floatnum to single-precision and return as 32-bit value.

The 32-bit value is a "standard" C value (eg, of unknown endian).

Parameters

flt	floatnum
ret_val	pointer to storage for 32-bit output

Returns

Nonzero if flt can't fit into single precision: -1 if underflow occurred, 1 if overflow occurred.

4.10.2.8 YASM_LIB_DECL int yasm_floatnum_get_sized (const yasm_floatnum * *flt,* unsigned char * *ptr,* size_t *destsize,* size_t *valsize,* size_t *shift,* int *bigendian,* int *warn*)

Output a yasm_floatnum to buffer in little-endian or big-endian.

Puts the value into the least significant bits of the destination, or may be shifted into more significant bits by the shift parameter. The destination bits are cleared before being set. [0] should be the first byte output to the file.

Note

Not all sizes are valid. Currently, only 32 (single-precision), 64 (double-precision), and 80 (extended-precision) are valid sizes. Use yasm_floatnum_check_size() to check for supported sizes.

Parameters

flt	floatnum
ptr	pointer to storage for size bytes of output
destsize	destination size (in bytes)
valsize	size (in bits)
shift	left shift (in bits)
bigendian	endianness (nonzero=big, zero=little)
warn	enables standard overflow/underflow warnings

Returns

Nonzero if flt can't fit into the specified precision: -1 if underflow occurred, 1 if overflow occurred.

4.10.2.9 YASM_LIB_DECL void yasm_floatnum_initialize (void)

Initialize floatnum internal data structures.

4.10.2.10 YASM_LIB_DECL void yasm_floatnum_print (const yasm_floatnum * *flt*, FILE * *f*)

Print various representations of a floatnum.

For debugging purposes only.

Parameters

f	file
flt	floatnum

4.11 libyasm/hamt.h File Reference

Hash Array Mapped Trie (HAMT) functions.

Typedefs

- typedef struct HAMT HAMT

  Hash array mapped trie data structure (opaque type).
- typedef struct HAMTEntry HAMTEntry

  Hash array mapped trie entry (opaque type).

Functions

- YASM_LIB_DECL HAMT * HAMT_create (int nocase, void(*error_func)(const char *file, unsigned int line, const char *message))

  Create new, empty, HAMT.
- YASM_LIB_DECL void HAMT_destroy (HAMT *hamt, void(*deletefunc)(void *data))

  Delete HAMT and all data associated with it.
- YASM_LIB_DECL void * HAMT_insert (HAMT *hamt, const char *str, void *data, int *replace, void(*deletefunc)(void *data))

  Insert key into HAMT, associating it with data.
- YASM_LIB_DECL void * HAMT_search (HAMT *hamt, const char *str)

  Search for the data associated with a key in the HAMT.
- YASM_LIB_DECL int HAMT_traverse (HAMT *hamt, void *d, int(*func)(void *node, void *d))

  Traverse over all keys in HAMT, calling function on each data item.
- YASM_LIB_DECL const HAMTEntry * HAMT_first (const HAMT *hamt)

  Get the first entry in a HAMT.
- YASM_LIB_DECL const HAMTEntry * HAMT_next (const HAMTEntry *prev)

  Get the next entry in a HAMT.
- YASM_LIB_DECL void * HAMTEntry_get_data (const HAMTEntry *entry)

  Get the corresponding data for a HAMT entry.

4.11.1 Detailed Description

Hash Array Mapped Trie (HAMT) functions.

Definition in file hamt.h.

4.11.2 Typedef Documentation

4.11.2.1 typedef struct HAMT HAMT

Hash array mapped trie data structure (opaque type).

Definition at line 38 of file hamt.h.

4.11.2.2 typedef struct HAMTEntry HAMTEntry

Hash array mapped trie entry (opaque type).

Definition at line 40 of file hamt.h.

4.11.3 Function Documentation

4.11.3.1 YASM_LIB_DECL HAMT* HAMT_create (int *nocase,* void(*)(const char *file, unsigned int line, const char *message) *error_func*)

Create new, empty, HAMT.

error_func() is called when an internal error is encountered–it should NOT return to the calling function.

Parameters

nocase	nonzero if HAMT should be case-insensitive
error_func	function called on internal error

Returns

New, empty, hash array mapped trie.

4.11.3.2 YASM_LIB_DECL void HAMT_destroy (HAMT * *hamt,* void(*)(void *data) *deletefunc*)

Delete HAMT and all data associated with it.

Uses deletefunc() to delete each data item.

Parameters

hamt	Hash array mapped trie
deletefunc	Data deletion function

4.11.3.3 YASM_LIB_DECL const HAMTEntry* HAMT_first (const HAMT * *hamt*)

Get the first entry in a HAMT.

Parameters

hamt	Hash array mapped trie

Returns

First entry in HAMT, or NULL if HAMT is empty.

4.11.3.4 YASM_LIB_DECL void* HAMT_insert (HAMT * *hamt,* const char * *str,* void * *data,* int * *replace,* void(*)(void *data) *deletefunc*)

Insert key into HAMT, associating it with data.

If the key is not present in the HAMT, inserts it, sets *replace to 1, and returns the data passed in. If the key is already present and *replace is 0, deletes the data passed in using deletefunc() and returns the data currently associated with the key. If the key is already present and *replace is 1, deletes the data currently associated with the key using deletefunc() and replaces it with the data passed in.

Parameters

hamt	Hash array mapped trie
str	Key
data	Data to associate with key
replace	See above description
deletefunc	Data deletion function if data is replaced

Returns

Data now associated with key.

4.11.3.5 YASM_LIB_DECL const HAMTEntry* HAMT_next (const HAMTEntry * *prev*)

Get the next entry in a HAMT.

Parameters

prev	Previous entry in HAMT

Returns

Next entry in HAMT, or NULL if no more entries.

4.11.3.6 YASM_LIB_DECL void* HAMT_search (HAMT * *hamt,* const char * *str*)

Search for the data associated with a key in the HAMT.

Parameters

hamt	Hash array mapped trie
str	Key

Returns

NULL if key/data not present in HAMT, otherwise associated data.

4.11.3.7 YASM_LIB_DECL int HAMT_traverse (HAMT * *hamt,* void * *d,* int(*)(void *node, void *d) *func*)

Traverse over all keys in HAMT, calling function on each data item.

Parameters

hamt	Hash array mapped trie
d	Data to pass to each call to func.
func	Function to call

Returns

Stops early (and returns func's return value) if func returns a nonzero value; otherwise 0.

4.11.3.8 YASM_LIB_DECL void* HAMTEntry_get_data (const HAMTEntry * *entry*)

Get the corresponding data for a HAMT entry.

Parameters

entry	HAMT entry (as returned by HAMT_first() and HAMT_next())

Returns

Corresponding data item.

4.12 libyasm/intnum.h File Reference

YASM integer number interface.

Functions

- YASM_LIB_DECL void yasm_intnum_initialize (void)

  Initialize intnum internal data structures.
- YASM_LIB_DECL void yasm_intnum_cleanup (void)

  Clean up internal intnum allocations.
- YASM_LIB_DECL yasm_intnum * yasm_intnum_create_dec (char *str)

  Create a new intnum from a decimal string.
- YASM_LIB_DECL yasm_intnum * yasm_intnum_create_bin (char *str)

  Create a new intnum from a binary string.
- YASM_LIB_DECL yasm_intnum * yasm_intnum_create_oct (char *str)

  Create a new intnum from an octal string.
- YASM_LIB_DECL yasm_intnum * yasm_intnum_create_hex (char *str)

  Create a new intnum from a hexidecimal string.
- YASM_LIB_DECL yasm_intnum * yasm_intnum_create_charconst_nasm (const char *str)

  Convert character constant to integer value, using NASM rules.
- YASM_LIB_DECL yasm_intnum * yasm_intnum_create_charconst_tasm (const char *str)

  Convert character constant to integer value, using TASM rules.
- YASM_LIB_DECL yasm_intnum * yasm_intnum_create_uint (unsigned long i)

  Create a new intnum from an unsigned integer value.
- YASM_LIB_DECL yasm_intnum * yasm_intnum_create_int (long i)

  Create a new intnum from an signed integer value.
- YASM_LIB_DECL yasm_intnum * yasm_intnum_create_leb128 (const unsigned char *ptr, int sign, unsigned long *size)

  Create a new intnum from LEB128-encoded form.
- YASM_LIB_DECL yasm_intnum * yasm_intnum_create_sized (unsigned char *ptr, int sign, size_t srcsize, int bigendian)

  Create a new intnum from a little-endian or big-endian buffer.
- YASM_LIB_DECL yasm_intnum * yasm_intnum_copy (const yasm_intnum *intn)

Duplicate an intnum.

- YASM_LIB_DECL void yasm_intnum_destroy (yasm_intnum *intn)

  Destroy (free allocated memory for) an intnum.
- YASM_LIB_DECL int yasm_intnum_calc (yasm_intnum *acc, yasm_expr_op op, yasm_intnum *operand)

  Floating point calculation function: acc = acc op operand.
- YASM_LIB_DECL int yasm_intnum_compare (const yasm_intnum *intn1, const yasm_intnum *intn2)

  Compare two intnums.
- YASM_LIB_DECL void yasm_intnum_zero (yasm_intnum *intn)

  Zero an intnum.
- YASM_LIB_DECL void yasm_intnum_set (yasm_intnum *intn, const yasm_intnum *val)

  Set an intnum to the value of another intnum.
- YASM_LIB_DECL void yasm_intnum_set_uint (yasm_intnum *intn, unsigned long val)

  Set an intnum to an unsigned integer.
- YASM_LIB_DECL void yasm_intnum_set_int (yasm_intnum *intn, long val)

  Set an intnum to an signed integer.
- YASM_LIB_DECL int yasm_intnum_is_zero (const yasm_intnum *acc)

  Simple value check for 0.
- YASM_LIB_DECL int yasm_intnum_is_pos1 (const yasm_intnum *acc)

  Simple value check for 1.
- YASM_LIB_DECL int yasm_intnum_is_neg1 (const yasm_intnum *acc)

  Simple value check for -1.
- YASM_LIB_DECL int yasm_intnum_sign (const yasm_intnum *acc)

  Simple sign check.
- YASM_LIB_DECL unsigned long yasm_intnum_get_uint (const yasm_intnum *intn)

  Convert an intnum to an unsigned 32-bit value.
- YASM_LIB_DECL long yasm_intnum_get_int (const yasm_intnum *intn)

  Convert an intnum to a signed 32-bit value.
- YASM_LIB_DECL void yasm_intnum_get_sized (const yasm_intnum *intn, unsigned char *ptr, size_t destsize, size_t valsize, int shift, int bigendian, int warn)

  Output yasm_intnum to buffer in little-endian or big-endian.
- YASM_LIB_DECL int yasm_intnum_check_size (const yasm_intnum *intn, size_t size, size_t rshift, int rangetype)

  Check to see if intnum will fit without overflow into size bits.
- YASM_LIB_DECL int yasm_intnum_in_range (const yasm_intnum *intn, long low, long high)

  Check to see if intnum will fit into a particular numeric range.
- YASM_LIB_DECL unsigned long yasm_intnum_get_leb128 (const yasm_intnum *intn, unsigned char *ptr, int sign)

  Output yasm_intnum to buffer in LEB128-encoded form.
- YASM_LIB_DECL unsigned long yasm_intnum_size_leb128 (const yasm_intnum *intn, int sign)

  Calculate number of bytes LEB128-encoded form of yasm_intnum will take.
- YASM_LIB_DECL unsigned long yasm_get_sleb128 (long v, unsigned char *ptr)

  Output integer to buffer in signed LEB128-encoded form.
- YASM_LIB_DECL unsigned long yasm_size_sleb128 (long v)

  Calculate number of bytes signed LEB128-encoded form of integer will take.
- YASM_LIB_DECL unsigned long yasm_get_uleb128 (unsigned long v, unsigned char *ptr)

  Output integer to buffer in unsigned LEB128-encoded form.
- YASM_LIB_DECL unsigned long yasm_size_uleb128 (unsigned long v)

Calculate number of bytes unsigned LEB128-encoded form of integer will take.

- YASM_LIB_DECL char * yasm_intnum_get_str (const yasm_intnum *intn)

  Get an intnum as a signed decimal string.

- YASM_LIB_DECL void yasm_intnum_print (const yasm_intnum *intn, FILE *f)

  Print an intnum.

4.12.1 Detailed Description

YASM integer number interface.

Definition in file intnum.h.

4.12.2 Function Documentation

4.12.2.1 YASM_LIB_DECL unsigned long yasm_get_sleb128 (long *v*, unsigned char * *ptr*)

Output integer to buffer in signed LEB128-encoded form.

Parameters

v	integer
ptr	pointer to storage for output bytes

Returns

Number of bytes generated.

4.12.2.2 YASM_LIB_DECL unsigned long yasm_get_uleb128 (unsigned long *v*, unsigned char * *ptr*)

Output integer to buffer in unsigned LEB128-encoded form.

Parameters

v	integer
ptr	pointer to storage for output bytes

Returns

Number of bytes generated.

4.12.2.3 YASM_LIB_DECL int yasm_intnum_calc (yasm_intnum * *acc*, yasm_expr_op *op*, yasm_intnum * *operand*)

Floating point calculation function: acc = acc op operand.

Note

Not all operations in yasm_expr_op may be supported; unsupported operations will result in an error.

Parameters

acc	intnum accumulator
op	operation
operand	intnum operand

Returns

Nonzero if error occurred.

4.12.2.4 YASM_LIB_DECL int yasm_intnum_check_size (const yasm_intnum * *intn*, size_t *size*, size_t *rshift*, int *rangetype*)

Check to see if intnum will fit without overflow into size bits.

Parameters

intn	intnum
size	number of bits of output space
rshift	right shift
rangetype	signed/unsigned range selection: 0 => (0, unsigned max); 1 => (signed min, signed max); 2 => (signed min, unsigned max)

Returns

Nonzero if intnum will fit.

4.12.2.5 YASM_LIB_DECL void yasm_intnum_cleanup (void)

Clean up internal intnum allocations.

4.12.2.6 YASM_LIB_DECL int yasm_intnum_compare (const yasm_intnum * *intn1*, const yasm_intnum * *intn2*)

Compare two intnums.

Parameters

intn1	first intnum
intn2	second intnum

Returns

-1 if intn1 $<$ intn2, 0 if intn1 == intn2, 1 if intn1 $>$ intn2.

4.12.2.7 YASM_LIB_DECL yasm_intnum* yasm_intnum_copy (const yasm_intnum * *intn*)

Duplicate an intnum.

Parameters

intn	intnum

Returns

Newly allocated intnum with the same value as intn.

4.12.2.8 YASM_LIB_DECL yasm_intnum* yasm_intnum_create_bin (char * *str*)

Create a new intnum from a binary string.

Parameters

str	binary string

Returns

Newly allocated intnum.

4.12.2.9 YASM_LIB_DECL yasm_intnum* yasm_intnum_create_charconst_nasm (const char * *str*)

Convert character constant to integer value, using NASM rules.

NASM syntax supports automatic conversion from strings such as 'abcd' to a 32-bit integer value (little endian order). This function performs those conversions.

Parameters

str	character constant string

Returns

Newly allocated intnum.

4.12.2.10 YASM_LIB_DECL yasm_intnum* yasm_intnum_create_charconst_tasm (const char * *str*)

Convert character constant to integer value, using TASM rules.

TASM syntax supports automatic conversion from strings such as 'abcd' to a 32-bit integer value (big endian order). This function performs those conversions.

Parameters

str	character constant string

Returns

Newly allocated intnum.

4.12.2.11 YASM_LIB_DECL yasm_intnum* yasm_intnum_create_dec (char * *str*)

Create a new intnum from a decimal string.

Parameters

str	decimal string

Returns

Newly allocated intnum.

4.12.2.12 YASM_LIB_DECL yasm_intnum* yasm_intnum_create_hex (char * *str*)

Create a new intnum from a hexidecimal string.

Parameters

str	hexidecimal string

Returns

Newly allocated intnum.

4.12.2.13 YASM_LIB_DECL yasm_intnum* yasm_intnum_create_int (long *i*)

Create a new intnum from an signed integer value.

Parameters

i	signed integer value

Returns

Newly allocated intnum.

4.12.2.14 YASM_LIB_DECL yasm_intnum* yasm_intnum_create_leb128 (const unsigned char * *ptr,* int *sign,* unsigned long * *size*)

Create a new intnum from LEB128-encoded form.

Parameters

ptr	pointer to start of LEB128 encoded form
sign	signed (1) or unsigned (0) LEB128 format
size	number of bytes read from ptr (output)

Returns

Newly allocated intnum. Number of bytes read returned into bytes_read parameter.

4.12.2.15 YASM_LIB_DECL yasm_intnum* yasm_intnum_create_oct (char * *str*)

Create a new intnum from an octal string.

Parameters

str	octal string

Returns

Newly allocated intnum.

4.12.2.16 YASM_LIB_DECL yasm_intnum* yasm_intnum_create_sized (unsigned char * *ptr,* int *sign,* size_t *srcsize,* int *bigendian*)

Create a new intnum from a little-endian or big-endian buffer.

In little endian, the LSB is in ptr[0].

Parameters

ptr	pointer to start of buffer
sign	signed (1) or unsigned (0) source
srcsize	source buffer size (in bytes)
bigendian	endianness (nonzero=big, zero=little)

4.12.2.17 YASM_LIB_DECL yasm_intnum* yasm_intnum_create_uint (unsigned long *i*)

Create a new intnum from an unsigned integer value.

Parameters

i	unsigned integer value

Returns

Newly allocated intnum.

4.12.2.18 YASM_LIB_DECL void yasm_intnum_destroy (yasm_intnum * *intn*)

Destroy (free allocated memory for) an intnum.

Parameters

intn	intnum

4.12.2.19 YASM_LIB_DECL long yasm_intnum_get_int (const yasm_intnum * *intn*)

Convert an intnum to a signed 32-bit value.

The value is in "standard" C format (eg, of unknown endian).

Note

Parameter intnum is truncated to fit into 32 bits. Use intnum_check_size() to check for overflow.

Parameters

intn	intnum

Returns

Signed 32-bit value of intn.

4.12.2.20 YASM_LIB_DECL unsigned long yasm_intnum_get_leb128 (const yasm_intnum * *intn,* unsigned char * *ptr,* int *sign*)

Output yasm_intnum to buffer in LEB128-encoded form.

Parameters

intn	intnum
ptr	pointer to storage for output bytes
sign	signedness of LEB128 encoding (0=unsigned, 1=signed)

Returns

Number of bytes generated.

4.12.2.21 YASM_LIB_DECL void yasm_intnum_get_sized (const yasm_intnum * *intn*, unsigned char * *ptr*, size_t *destsize*, size_t *valsize*, int *shift*, int *bigendian*, int *warn*)

Output yasm_intnum to buffer in little-endian or big-endian.

Puts the value into the least significant bits of the destination, or may be shifted into more significant bits by the shift parameter. The destination bits are cleared before being set. [0] should be the first byte output to the file.

Parameters

intn	intnum
ptr	pointer to storage for size bytes of output
destsize	destination size (in bytes)
valsize	size (in bits)
shift	left shift (in bits); may be negative to specify right shift (standard warnings include truncation to boundary)
bigendian	endianness (nonzero=big, zero=little)
warn	enables standard warnings (value doesn't fit into valsize bits): <0=signed warnings, >0=unsigned warnings, 0=no warn

4.12.2.22 YASM_LIB_DECL char* yasm_intnum_get_str (const yasm_intnum * *intn*)

Get an intnum as a signed decimal string.

The returned string will contain a leading '-' if the intnum is negative.

Parameters

intn	intnum

Returns

Newly allocated string containing the decimal representation of the intnum.

4.12.2.23 YASM_LIB_DECL unsigned long yasm_intnum_get_uint (const yasm_intnum * *intn*)

Convert an intnum to an unsigned 32-bit value.

The value is in "standard" C format (eg, of unknown endian).

Note

Parameter intnum is truncated to fit into 32 bits. Use intnum_check_size() to check for overflow.

Parameters

intn	intnum

Returns

Unsigned 32-bit value of intn.

4.12.2.24 YASM_LIB_DECL int yasm_intnum_in_range (const yasm_intnum * *intn,* long *low,* long *high*)

Check to see if intnum will fit into a particular numeric range.

Parameters

intn	intnum
low	low end of range (inclusive)
high	high end of range (inclusive)

Returns

Nonzero if intnum is within range.

4.12.2.25 YASM_LIB_DECL void yasm_intnum_initialize (void)

Initialize intnum internal data structures.

4.12.2.26 YASM_LIB_DECL int yasm_intnum_is_neg1 (const yasm_intnum * *acc*)

Simple value check for -1.

Parameters

acc	intnum

Returns

Nonzero if acc==-1.

4.12.2.27 YASM_LIB_DECL int yasm_intnum_is_pos1 (const yasm_intnum * *acc*)

Simple value check for 1.

Parameters

acc	intnum

Returns

Nonzero if acc==1.

4.12.2.28 YASM_LIB_DECL int yasm_intnum_is_zero (const yasm_intnum * *acc*)

Simple value check for 0.

Parameters

acc	intnum

Returns

Nonzero if acc==0.

4.12.2.29 YASM_LIB_DECL void yasm_intnum_print (const yasm_intnum * intn, FILE * f)

Print an intnum.

For debugging purposes.

Parameters

f	file
intn	intnum

4.12.2.30 YASM_LIB_DECL void yasm_intnum_set (yasm_intnum * intn, const yasm_intnum * val)

Set an intnum to the value of another intnum.

Parameters

intn	intnum
val	intnum to get value from

4.12.2.31 YASM_LIB_DECL void yasm_intnum_set_int (yasm_intnum * intn, long val)

Set an intnum to an signed integer.

Parameters

intn	intnum
val	integer value

4.12.2.32 YASM_LIB_DECL void yasm_intnum_set_uint (yasm_intnum * intn, unsigned long val)

Set an intnum to an unsigned integer.

Parameters

intn	intnum
val	integer value

4.12.2.33 YASM_LIB_DECL int yasm_intnum_sign (const yasm_intnum * acc)

Simple sign check.

Parameters

acc	intnum

Returns

-1 if negative, 0 if zero, +1 if positive

4.12.2.34 YASM_LIB_DECL unsigned long yasm_intnum_size_leb128 (const yasm_intnum * intn, int sign)

Calculate number of bytes LEB128-encoded form of yasm_intnum will take.

Parameters

intn	intnum
sign	signedness of LEB128 encoding (0=unsigned, 1=signed)

Returns

Number of bytes.

4.12.2.35 YASM_LIB_DECL void yasm_intnum_zero (yasm_intnum * *intn*)

Zero an intnum.

Parameters

intn	intnum

4.12.2.36 YASM_LIB_DECL unsigned long yasm_size_sleb128 (long *v*)

Calculate number of bytes signed LEB128-encoded form of integer will take.

Parameters

v	integer

Returns

Number of bytes.

4.12.2.37 YASM_LIB_DECL unsigned long yasm_size_uleb128 (unsigned long *v*)

Calculate number of bytes unsigned LEB128-encoded form of integer will take.

Parameters

v	integer

Returns

Number of bytes.

4.13 libyasm/linemap.h File Reference

YASM virtual line mapping interface.

Functions

- YASM_LIB_DECL yasm_linemap * yasm_linemap_create (void)

  Create a new line mapping repository.
- YASM_LIB_DECL void yasm_linemap_destroy (yasm_linemap *linemap)

  Clean up any memory allocated for a repository.
- YASM_LIB_DECL unsigned long yasm_linemap_get_current (yasm_linemap *linemap)

Get the current line position in a repository.

- YASM_LIB_DECL int yasm_linemap_get_source (yasm_linemap *linemap, unsigned long line, yasm_bytecode **bcp, const char **sourcep)

  Get bytecode and source line information, if any, for a virtual line.
- YASM_LIB_DECL void yasm_linemap_add_source (yasm_linemap *linemap, yasm_bytecode *bc, const char *source)

  Add bytecode and source line information to the current virtual line.
- YASM_LIB_DECL unsigned long yasm_linemap_goto_next (yasm_linemap *linemap)

  Go to the next line (increments the current virtual line).
- YASM_LIB_DECL void yasm_linemap_set (yasm_linemap *linemap, const char *filename, unsigned long virtual_line, unsigned long file_line, unsigned long line_inc)

  Set a new file/line physical association starting point at the specified virtual line.
- YASM_LIB_DECL unsigned long yasm_linemap_poke (yasm_linemap *linemap, const char *filename, unsigned long file_line)

  Poke a single file/line association, restoring the original physical association starting point.
- YASM_LIB_DECL void yasm_linemap_lookup (yasm_linemap *linemap, unsigned long line, const char **filename, unsigned long *file_line)

  Look up the associated physical file and line for a virtual line.
- YASM_LIB_DECL int yasm_linemap_traverse_filenames (yasm_linemap *linemap, void *d, int(*func)(const char *filename, void *d))

  Traverses all filenames used in a linemap, calling a function on each filename.

4.13.1 Detailed Description

YASM virtual line mapping interface.

Definition in file linemap.h.

4.13.2 Function Documentation

4.13.2.1 YASM_LIB_DECL void yasm_linemap_add_source (yasm_linemap * *linemap*, yasm_bytecode * *bc*, const char * *source*)

Add bytecode and source line information to the current virtual line.

Attention

Deletes any existing bytecode and source line information for the current virtual line.

Parameters

linemap	line mapping repository
bc	bytecode (if any)
source	source code line

Note

The source code line pointer is NOT kept, it is strdup'ed.

4.13.2.2 YASM_LIB_DECL yasm_linemap* yasm_linemap_create (void)

Create a new line mapping repository.

Returns

New repository.

4.13.2.3 YASM_LIB_DECL void yasm_linemap_destroy (yasm_linemap * *linemap*)

Clean up any memory allocated for a repository.

Parameters

linemap	line mapping repository

4.13.2.4 YASM_LIB_DECL unsigned long yasm_linemap_get_current (yasm_linemap * *linemap*)

Get the current line position in a repository.

Parameters

linemap	line mapping repository

Returns

Current virtual line.

4.13.2.5 YASM_LIB_DECL int yasm_linemap_get_source (yasm_linemap * *linemap,* unsigned long *line,* yasm_bytecode ** *bcp,* const char ** *sourcep*)

Get bytecode and source line information, if any, for a virtual line.

Parameters

linemap	line mapping repository
line	virtual line
bcp	pointer to return bytecode into
sourcep	pointer to return source code line pointer into

Returns

Zero if source line information available for line, nonzero if not.

Note

If source line information is not available, bcp and sourcep targets are set to NULL.

4.13.2.6 YASM_LIB_DECL unsigned long yasm_linemap_goto_next (yasm_linemap * *linemap*)

Go to the next line (increments the current virtual line).

Parameters

linemap	line mapping repository

Returns

The current (new) virtual line.

4.13.2.7 YASM_LIB_DECL void yasm_linemap_lookup (yasm_linemap * *linemap,* unsigned long *line,* const char ** *filename,* unsigned long * *file_line*)

Look up the associated physical file and line for a virtual line.

Parameters

linemap	line mapping repository
line	virtual line
filename	physical file name (output)
file_line	physical line number (output)

4.13.2.8 YASM_LIB_DECL unsigned long yasm_linemap_poke (yasm_linemap * *linemap,* const char * *filename,* unsigned long *file_line*)

Poke a single file/line association, restoring the original physical association starting point.

Caution: increments the current virtual line twice.

Parameters

linemap	line mapping repository
filename	physical file name (if NULL, not changed)
file_line	physical line number

Returns

The virtual line number of the poked association.

4.13.2.9 YASM_LIB_DECL void yasm_linemap_set (yasm_linemap * *linemap,* const char * *filename,* unsigned long *virtual_line,* unsigned long *file_line,* unsigned long *line_inc*)

Set a new file/line physical association starting point at the specified virtual line.

line_inc indicates how much the "real" line is incremented by for each virtual line increment (0 is perfectly legal).

Parameters

linemap	line mapping repository
filename	physical file name (if NULL, not changed)
virtual_line	virtual line number (if 0, linemap->current is used)
file_line	physical line number
line_inc	line increment

4.13.2.10 YASM_LIB_DECL int yasm_linemap_traverse_filenames (yasm_linemap * *linemap*, void * *d*, int(*)(const char *filename, void *d) *func*)

Traverses all filenames used in a linemap, calling a function on each filename.

Parameters

linemap	line mapping repository
d	data pointer passed to func on each call
func	function

Returns

Stops early (and returns func's return value) if func returns a nonzero value; otherwise 0.

4.14 libyasm/listfmt.h File Reference

YASM list format interface.

Data Structures

- struct yasm_listfmt_module

  YASM list format module interface.

Typedefs

- typedef struct yasm_listfmt_module yasm_listfmt_module

  YASM list format module interface.

Functions

- const char * yasm_listfmt_keyword (const yasm_listfmt *listfmt)

  Get the keyword used to select a list format.
- yasm_listfmt * yasm_listfmt_create (const yasm_listfmt_module *module, const char *in_filename, const char *obj_filename)

  Initialize list format for use.
- void yasm_listfmt_destroy (yasm_listfmt *listfmt)

  Cleans up any allocated list format memory.
- void yasm_listfmt_output (yasm_listfmt *listfmt, FILE *f, yasm_linemap *linemap, yasm_arch *arch)

  Write out list to the list file.

4.14.1 Detailed Description

YASM list format interface.

Definition in file listfmt.h.

4.14.2 Typedef Documentation

4.14.2.1 typedef struct yasm_listfmt_module yasm_listfmt_module

YASM list format module interface.

4.14.3 Function Documentation

4.14.3.1 yasm_listfmt* yasm_listfmt_create (const yasm_listfmt_module * *module*, const char * *in_filename*, const char * *obj_filename*)

Initialize list format for use.

Must call before any other list format functions. The filenames are provided solely for informational purposes.

Parameters

module	list format module
in_filename	primary input filename
obj_filename	object filename

Returns

NULL if object format does not provide needed support.

4.14.3.2 void yasm_listfmt_destroy (yasm_listfmt * *listfmt*)

Cleans up any allocated list format memory.

Parameters

listfmt	list format

4.14.3.3 const char* yasm_listfmt_keyword (const yasm_listfmt * *listfmt*)

Get the keyword used to select a list format.

Parameters

listfmt	list format

Returns

keyword

4.14.3.4 void yasm_listfmt_output (yasm_listfmt * *listfmt*, FILE * *f*, yasm_linemap * *linemap*, yasm_arch * *arch*)

Write out list to the list file.

This function may call all read-only yasm_* functions as necessary.

Parameters

listfmt	list format
f	output list file
linemap	line mapping repository
arch	architecture

4.15 libyasm/objfmt.h File Reference

YASM object format module interface.

Data Structures

- struct yasm_objfmt_module

  Object format module interface.

Functions

- yasm_objfmt * yasm_objfmt_create (const yasm_objfmt_module *module, yasm_object *object)

  Create object format.
- void yasm_objfmt_output (yasm_object *object, FILE *f, int all_syms, yasm_errwarns *errwarns)

  Write out (post-optimized) sections to the object file.
- void yasm_objfmt_destroy (yasm_objfmt *objfmt)

  Cleans up any allocated object format memory.
- yasm_section * yasm_objfmt_add_default_section (yasm_object *object)

  Add a default section to an object.
- void yasm_objfmt_init_new_section (yasm_object *object, unsigned long line)

  Initialize the object-format specific portion of a section.
- yasm_section * yasm_objfmt_section_switch (yasm_object *object, yasm_valparamhead *valparams, yasm_valparamhead *objext_valparams, unsigned long line)

  Switch object file sections.
- yasm_symrec * yasm_objfmt_get_special_sym (yasm_object *object, const char *name, const char *parser)

  Get a special symbol.

4.15.1 Detailed Description

YASM object format module interface.

Definition in file objfmt.h.

4.15.2 Function Documentation

4.15.2.1 yasm_section* yasm_objfmt_add_default_section (yasm_object * *object*)

Add a default section to an object.

Parameters

object	object

Returns

Default section.

4.15.2.2 yasm_objfmt* yasm_objfmt_create (const yasm_objfmt_module * *module*, yasm_object * *object*)

Create object format.

Parameters

module	object format module
object	object

Returns

NULL if architecture/machine combination not supported.

4.15.2.3 void yasm_objfmt_destroy (yasm_objfmt * *objfmt*)

Cleans up any allocated object format memory.

Parameters

objfmt	object format

4.15.2.4 yasm_symrec* yasm_objfmt_get_special_sym (yasm_object * *object*, const char * *name*, const char * *parser*)

Get a special symbol.

Special symbols are generally used to generate special relocation types via the WRT mechanism.

Parameters

object	object
name	symbol name (not including any parser-specific prefix)
parser	parser keyword

Returns

NULL if unrecognized, otherwise special symbol.

4.15.2.5 void yasm_objfmt_init_new_section (yasm_object * *object*, unsigned long *line*)

Initialize the object-format specific portion of a section.

Called by yasm_object_get_general(); in general should not be directly called.

Parameters

section	section
line	virtual line (from yasm_linemap)

4.15.2.6 void yasm_objfmt_output (yasm_object * *object*, FILE * *f*, int *all_syms*, yasm_errwarns * *errwarns*)

Write out (post-optimized) sections to the object file.

This function may call yasm_symrec_* functions as necessary (including yasm_symrec_traverse()) to retrieve symbolic information.

Parameters

object	object
f	output object file
all_syms	if nonzero, all symbols should be included in the object file
errwarns	error/warning set

Note

Errors and warnings are stored into errwarns.

4.15.2.7 yasm_section* yasm_objfmt_section_switch (yasm_object * *object*, yasm_valparamhead * *valparams*, yasm_valparamhead * *objext_valparams*, unsigned long *line*)

Switch object file sections.

The first val of the valparams should be the section name. Calls yasm_object_get_general() to actually get the section.

Parameters

object	object
valparams	value/parameters
objext_valparams	object format-specific value/parameters
line	virtual line (from yasm_linemap)

Returns

NULL on error, otherwise new section.

4.16 libyasm/parser.h File Reference

YASM parser module interface.

Data Structures

- struct yasm_parser_module

  YASM parser module interface.

Typedefs

- typedef struct yasm_parser_module yasm_parser_module

  YASM parser module interface.

4.16.1 Detailed Description

YASM parser module interface.

Definition in file parser.h.

4.16.2 Typedef Documentation

4.16.2.1 typedef struct yasm_parser_module yasm_parser_module

YASM parser module interface.

The "front end" of the assembler.

4.17 libyasm/preproc.h File Reference

YASM preprocessor module interface.

Data Structures

- struct yasm_preproc_module
  YASM preprocesor module interface.

Typedefs

- typedef struct yasm_preproc_module yasm_preproc_module
  YASM preprocesor module interface.

Functions

- yasm_preproc * yasm_preproc_create (yasm_preproc_module *module, const char *in_filename, yasm_symtab *symtab, yasm_linemap *lm, yasm_errwarns *errwarns)
  Initialize preprocessor.
- void yasm_preproc_destroy (yasm_preproc *preproc)
  Cleans up any allocated preproc memory.
- char * yasm_preproc_get_line (yasm_preproc *preproc)
  Gets a single line of preprocessed source code.
- size_t yasm_preproc_get_included_file (yasm_preproc *preproc, char *buf, size_t max_size)
  Get the next filename included by the source code.
- void yasm_preproc_add_include_file (yasm_preproc *preproc, const char *filename)
  Pre-include a file.
- void yasm_preproc_predefine_macro (yasm_preproc *preproc, const char *macronameval)
  Pre-define a macro.
- void yasm_preproc_undefine_macro (yasm_preproc *preproc, const char *macroname)
  Un-define a macro.
- void yasm_preproc_define_builtin (yasm_preproc *preproc, const char *macronameval)
  Define a builtin macro, preprocessed before the "standard" macros.
- void yasm_preproc_add_standard (yasm_preproc *preproc, const char **macros)
  Define additional standard macros, preprocessed after the builtins but prior to any user-defined macros.

4.17.1 Detailed Description

YASM preprocessor module interface.

Definition in file preproc.h.

4.17.2 Typedef Documentation

4.17.2.1 typedef struct yasm_preproc_module yasm_preproc_module

YASM preprocesor module interface.

4.17.3 Function Documentation

4.17.3.1 void yasm_preproc_add_include_file (yasm_preproc * *preproc,* const char * *filename*)

Pre-include a file.

Parameters

preproc	preprocessor
filename	filename

4.17.3.2 void yasm_preproc_add_standard (yasm_preproc * *preproc,* const char ** *macros*)

Define additional standard macros, preprocessed after the builtins but prior to any user-defined macros.

Parameters

preproc	preprocessor
macros	NULL-terminated array of macro strings

4.17.3.3 yasm_preproc* yasm_preproc_create (yasm_preproc_module * *module,* const char * *in_filename,* yasm_symtab * *symtab,* yasm_linemap * *lm,* yasm_errwarns * *errwarns*)

Initialize preprocessor.

The preprocessor needs access to the object format module to find out any output format specific macros.

Parameters

module	preprocessor module
in_filename	initial starting filename, or "-" to read from stdin
symtab	symbol table (may be NULL if none)
lm	line mapping repository
errwarns	error/warning set

Returns

New preprocessor.

Note

Errors/warnings are stored into errwarns.

4.17.3.4 void yasm_preproc_define_builtin (yasm_preproc * preproc, const char * macronameval)

Define a builtin macro, preprocessed before the "standard" macros.

Parameters

preproc	preprocessor
macronameval	"name=value" string

4.17.3.5 void yasm_preproc_destroy (yasm_preproc * preproc)

Cleans up any allocated preproc memory.

Parameters

preproc	preprocessor

4.17.3.6 size_t yasm_preproc_get_included_file (yasm_preproc * preproc, char * buf, size_t max_size)

Get the next filename included by the source code.

Parameters

preproc	preprocessor
buf	destination buffer for filename
max_size	maximum number of bytes that can be returned in buf

Returns

Actual number of bytes returned in buf.

4.17.3.7 char* yasm_preproc_get_line (yasm_preproc * preproc)

Gets a single line of preprocessed source code.

Parameters

preproc	preprocessor

Returns

Allocated line of code, without the trailing
.

4.17.3.8 void yasm_preproc_predefine_macro (yasm_preproc * preproc, const char * macronameval)

Pre-define a macro.

Parameters

preproc	preprocessor
macronameval	"name=value" string

4.17.3.9 void yasm_preproc_undefine_macro (yasm_preproc * *preproc,* const char * *macroname*)

Un-define a macro.

Parameters

preproc	preprocessor
macroname	macro name

4.18 libyasm/section.h File Reference

YASM section interface.

Data Structures

- struct yasm_reloc
- struct yasm_overrides

  Structure of functions that can be overridden.
- struct yasm_object

  An object.

Typedefs

- typedef struct yasm_reloc yasm_reloc

  Basic YASM relocation.
- typedef struct yasm_overrides yasm_overrides

  Structure of functions that can be overridden.

Functions

- YASM_LIB_DECL yasm_object * yasm_object_create (const char *src_filename, const char *obj_filename, yasm_arch *arch, const yasm_objfmt_module *objfmt_module, const yasm_dbgfmt_module *dbgfmt_module)

  Create a new object.
- YASM_LIB_DECL yasm_section * yasm_object_get_general (yasm_object *object, const char *name, unsigned long align, int code, int res_only, int *isnew, unsigned long line)

  Create a new, or continue an existing, general section.
- YASM_LIB_DECL int yasm_object_directive (yasm_object *object, const char *name, const char *parser, yasm_valparamhead *valparams, yasm_valparamhead *objext_valparams, unsigned long line)

  Handle a directive.
- YASM_LIB_DECL void yasm_object_destroy (yasm_object *object)

  Delete (free allocated memory for) an object.
- YASM_LIB_DECL void yasm_object_print (const yasm_object *object, FILE *f, int indent_level)

  Print an object.
- YASM_LIB_DECL void yasm_object_finalize (yasm_object *object, yasm_errwarns *errwarns)

  Finalize an object after parsing.
- YASM_LIB_DECL int yasm_object_sections_traverse (yasm_object *object, void *d, int(*func)(yasm_section *sect, void *d))

  Traverses all sections in an object, calling a function on each section.
- YASM_LIB_DECL yasm_section * yasm_object_find_general (yasm_object *object, const char *name)

Find a general section in an object, based on its name.

- YASM_LIB_DECL void yasm_object_set_source_fn (yasm_object *object, const char *src_filename)

  Change the source filename for an object.
- YASM_LIB_DECL void yasm_object_set_global_prefix (yasm_object *object, const char *prefix)

  Change the prefix used for externally-visible symbols.
- YASM_LIB_DECL void yasm_object_set_global_suffix (yasm_object *object, const char *suffix)

  Change the suffix used for externally-visible symbols.
- YASM_LIB_DECL void yasm_object_optimize (yasm_object *object, yasm_errwarns *errwarns)

  Optimize an object.
- YASM_LIB_DECL int yasm_section_is_code (yasm_section *sect)

  Determine if a section is flagged to contain code.
- YASM_LIB_DECL unsigned long yasm_section_get_opt_flags (const yasm_section *sect)

  Get yasm_optimizer-specific flags.
- YASM_LIB_DECL void yasm_section_set_opt_flags (yasm_section *sect, unsigned long opt_flags)

  Set yasm_optimizer-specific flags.
- YASM_LIB_DECL int yasm_section_is_default (const yasm_section *sect)

  Determine if a section was declared as the "default" section (e.g.
- YASM_LIB_DECL void yasm_section_set_default (yasm_section *sect, int def)

  Set section "default" flag to a new value.
- YASM_LIB_DECL yasm_object * yasm_section_get_object (const yasm_section *sect)

  Get object owner of a section.
- YASM_LIB_DECL void * yasm_section_get_data (yasm_section *sect, const yasm_assoc_data_callback *callback)

  Get assocated data for a section and data callback.
- YASM_LIB_DECL void yasm_section_add_data (yasm_section *sect, const yasm_assoc_data_callback *callback, void *data)

  Add associated data to a section.
- YASM_LIB_DECL void yasm_section_add_reloc (yasm_section *sect, yasm_reloc *reloc, void(*destroy_-func)(void *reloc))

  Add a relocation to a section.
- YASM_LIB_DECL yasm_reloc * yasm_section_relocs_first (yasm_section *sect)

  Get the first relocation for a section.
- yasm_reloc * yasm_section_reloc_next (yasm_reloc *reloc)

  Get the next relocation for a section.
- YASM_LIB_DECL void yasm_reloc_get (yasm_reloc *reloc, yasm_intnum **addrp, yasm_symrec **symp)

  Get the basic relocation information for a relocation.
- YASM_LIB_DECL yasm_bytecode * yasm_section_bcs_first (yasm_section *sect)

  Get the first bytecode in a section.
- YASM_LIB_DECL yasm_bytecode * yasm_section_bcs_last (yasm_section *sect)

  Get the last bytecode in a section.
- YASM_LIB_DECL yasm_bytecode * yasm_section_bcs_append (yasm_section *sect, yasm_bytecode *bc)

  Add bytecode to the end of a section.
- YASM_LIB_DECL int yasm_section_bcs_traverse (yasm_section *sect, yasm_errwarns *errwarns, void *d, int(*func)(yasm_bytecode *bc, void *d))

  Traverses all bytecodes in a section, calling a function on each bytecode.
- YASM_LIB_DECL const char * yasm_section_get_name (const yasm_section *sect)

  Get name of a section.
- YASM_LIB_DECL void yasm_section_set_align (yasm_section *sect, unsigned long align, unsigned long line)

Change alignment of a section.

- YASM_LIB_DECL unsigned long yasm_section_get_align (const yasm_section *sect)

  Get alignment of a section.

- YASM_LIB_DECL void yasm_section_print (const yasm_section *sect, FILE *f, int indent_level, int print_bcs)

  Print a section.

4.18.1 Detailed Description

YASM section interface.

Definition in file section.h.

4.18.2 Typedef Documentation

4.18.2.1 typedef struct yasm_reloc yasm_reloc

Basic YASM relocation.

Object formats will need to extend this structure with additional fields for relocation type, etc.

Definition at line 40 of file section.h.

4.18.3 Function Documentation

4.18.3.1 YASM_LIB_DECL yasm_object* yasm_object_create (const char * *src_filename*, const char * *obj_filename*, yasm_arch * *arch*, const yasm_objfmt_module * *objfmt_module*, const yasm_dbgfmt_module * *dbgfmt_module*)

Create a new object.

A default section is created as the first section. An empty symbol table (yasm_symtab) and line mapping (yasm_linemap) are automatically created.

Parameters

src_filename	source filename (e.g. "file.asm")
obj_filename	object filename (e.g. "file.o")
arch	architecture
objfmt_module	object format module
dbgfmt_module	debug format module

Returns

Newly allocated object, or NULL on error.

4.18.3.2 YASM_LIB_DECL void yasm_object_destroy (yasm_object * *object*)

Delete (free allocated memory for) an object.

All sections in the object and all bytecodes within those sections are also deleted.

Parameters

object	object

4.18.3.3 YASM_LIB_DECL int yasm_object_directive (yasm_object * *object*, const char * *name*, const char * *parser*, yasm_valparamhead * *valparams*, yasm_valparamhead * *objext_valparams*, unsigned long *line*)

Handle a directive.

Passed down to object format, debug format, or architecture as appropriate.

Parameters

object	object
name	directive name
parser	parser keyword
valparams	value/parameters
objext_valparams	"object format-specific" value/parameters
line	virtual line (from yasm_linemap)

Returns

0 if directive recognized, nonzero if unrecognized.

4.18.3.4 YASM_LIB_DECL void yasm_object_finalize (yasm_object * *object*, yasm_errwarns * *errwarns*)

Finalize an object after parsing.

Parameters

object	object
errwarns	error/warning set

Note

Errors/warnings are stored into errwarns.

4.18.3.5 YASM_LIB_DECL yasm_section* yasm_object_find_general (yasm_object * *object*, const char * *name*)

Find a general section in an object, based on its name.

Parameters

object	object
name	section name

Returns

Section matching name, or NULL if no match found.

4.18.3.6 YASM_LIB_DECL yasm_section* yasm_object_get_general (yasm_object * *object*, const char * *name*, unsigned long *align*, int *code*, int *res_only*, int * *isnew*, unsigned long *line*)

Create a new, or continue an existing, general section.

The section is added to the object if there's not already a section by that name.

Parameters

object	object
name	section name
align	alignment in bytes (0 if none)
code	if nonzero, section is intended to contain code (e.g. alignment should be made with NOP instructions, not 0)
res_only	if nonzero, only space-reserving bytecodes are allowed in the section (ignored if section already exists)
isnew	output; set to nonzero if section did not already exist
line	virtual line of section declaration (ignored if section already exists)

Returns

New section.

4.18.3.7 YASM_LIB_DECL void yasm_object_optimize (yasm_object * *object*, yasm_errwarns * *errwarns*)

Optimize an object.

Takes the unoptimized object and optimizes it. If successful, the object is ready for output to an object file.

Parameters

object	object
errwarns	error/warning set

Note

Optimization failures are stored into errwarns.

4.18.3.8 YASM_LIB_DECL void yasm_object_print (const yasm_object * *object*, FILE * *f*, int *indent_level*)

Print an object.

For debugging purposes.

Parameters

object	object
f	file
indent_level	indentation level

4.18.3.9 YASM_LIB_DECL int yasm_object_sections_traverse (yasm_object * *object*, void * *d*, int(*)(yasm_section *sect, void *d) *func*)

Traverses all sections in an object, calling a function on each section.

Parameters

object	object
d	data pointer passed to func on each call
func	function

Returns

Stops early (and returns func's return value) if func returns a nonzero value; otherwise 0.

4.18.3.10 YASM_LIB_DECL void yasm_object_set_global_prefix (yasm_object * *object*, const char * *prefix*)

Change the prefix used for externally-visible symbols.

Parameters

object	object
prefix	new prefix

4.18.3.11 YASM_LIB_DECL void yasm_object_set_global_suffix (yasm_object * *object*, const char * *suffix*)

Change the suffix used for externally-visible symbols.

Parameters

object	object
suffix	new suffix

4.18.3.12 YASM_LIB_DECL void yasm_object_set_source_fn (yasm_object * *object*, const char * *src_filename*)

Change the source filename for an object.

Parameters

object	object
src_filename	new source filename (e.g. "file.asm")

4.18.3.13 YASM_LIB_DECL void yasm_reloc_get (yasm_reloc * *reloc*, yasm_intnum ** *addrp*, yasm_symrec ** *symp*)

Get the basic relocation information for a relocation.

Parameters

reloc	relocation
addrp	address of relocation within section (returned)
symp	relocated symbol (returned)

4.18.3.14 YASM_LIB_DECL void yasm_section_add_data (yasm_section * *sect*, const yasm_assoc_data_callback * *callback*, void * *data*)

Add associated data to a section.

Attention

Deletes any existing associated data for that data callback.

Parameters

sect	section
callback	callback
data	data to associate

4.18.3.15 YASM_LIB_DECL void yasm_section_add_reloc (yasm_section * *sect,* yasm_reloc * *reloc,* void(*)(void *reloc) *destroy_func*)

Add a relocation to a section.

Parameters

sect	section
reloc	relocation
destroy_func	function that can destroy the relocation

Note

Does not make a copy of reloc. The same destroy_func must be used for all relocations in a section or an internal error will occur. The section will destroy the relocation address; it is the caller's responsibility to destroy any other allocated data.

4.18.3.16 YASM_LIB_DECL yasm_bytecode* yasm_section_bcs_append (yasm_section * *sect,* yasm_bytecode * *bc*)

Add bytecode to the end of a section.

Note

Does not make a copy of bc; so don't pass this function static or local variables, and discard the bc pointer after calling this function.

Parameters

sect	section
bc	bytecode (may be NULL)

Returns

If bytecode was actually appended (it wasn't NULL or empty), the bytecode; otherwise NULL.

4.18.3.17 YASM_LIB_DECL yasm_bytecode* yasm_section_bcs_first (yasm_section * *sect*)

Get the first bytecode in a section.

Parameters

sect	section

Returns

First bytecode in section (at least one empty bytecode is always present).

4.18.3.18 YASM_LIB_DECL yasm_bytecode* yasm_section_bcs_last (yasm_section * *sect*)

Get the last bytecode in a section.

Parameters

sect	section

Returns

Last bytecode in section (at least one empty bytecode is always present).

4.18.3.19 YASM_LIB_DECL int yasm_section_bcs_traverse (yasm_section * *sect*, yasm_errwarns * *errwarns*, void * *d*, int(*)(yasm_bytecode *bc, void *d) *func*)

Traverses all bytecodes in a section, calling a function on each bytecode.

Parameters

sect	section
errwarns	error/warning set (may be NULL)
d	data pointer passed to func on each call (may be NULL)
func	function

Returns

Stops early (and returns func's return value) if func returns a nonzero value; otherwise 0.

Note

If errwarns is non-NULL, yasm_errwarn_propagate() is called after each call to func (with the bytecode's line number).

4.18.3.20 YASM_LIB_DECL unsigned long yasm_section_get_align (const yasm_section * *sect*)

Get alignment of a section.

Parameters

sect	section

Returns

Alignment in bytes (0 if none).

4.18.3.21 YASM_LIB_DECL void* yasm_section_get_data (yasm_section * *sect*, const yasm_assoc_data_callback * *callback*)

Get assocated data for a section and data callback.

Parameters

sect	section
callback	callback used when adding data

Returns

Associated data (NULL if none).

4.18.3.22 YASM_LIB_DECL const char* yasm_section_get_name (const yasm_section * *sect*)

Get name of a section.

Parameters

sect	section

Returns

Section name.

4.18.3.23 YASM_LIB_DECL yasm_object* yasm_section_get_object (const yasm_section * *sect*)

Get object owner of a section.

Parameters

sect	section

Returns

Object this section is a part of.

4.18.3.24 YASM_LIB_DECL unsigned long yasm_section_get_opt_flags (const yasm_section * *sect*)

Get yasm_optimizer-specific flags.

For yasm_optimizer use only.

Parameters

sect	section

Returns

Optimizer-specific flags.

4.18.3.25 YASM_LIB_DECL int yasm_section_is_code (yasm_section * *sect*)

Determine if a section is flagged to contain code.

Parameters

sect	section

Returns

Nonzero if section is flagged to contain code.

4.18.3.26 YASM_LIB_DECL int yasm_section_is_default (const yasm_section * *sect*)

Determine if a section was declared as the "default" section (e.g.

not created through a section directive).

Parameters

sect	section

Returns

Nonzero if section was declared as default.

4.18.3.27 YASM_LIB_DECL void yasm_section_print (const yasm_section * *sect,* FILE * *f,* int *indent_level,* int *print_bcs*)

Print a section.

For debugging purposes.

Parameters

f	file
indent_level	indentation level
sect	section
print_bcs	if nonzero, print bytecodes within section

4.18.3.28 yasm_reloc* yasm_section_reloc_next (yasm_reloc * *reloc*)

Get the next relocation for a section.

Parameters

reloc	previous relocation

Returns

Next relocation for section. NULL if no more relocations.

4.18.3.29 YASM_LIB_DECL yasm_reloc* yasm_section_relocs_first (yasm_section * *sect*)

Get the first relocation for a section.

Parameters

sect	section

Returns

First relocation for section. NULL if no relocations.

4.18.3.30 YASM_LIB_DECL void yasm_section_set_align (yasm_section * *sect,* unsigned long *align,* unsigned long *line*)

Change alignment of a section.

Parameters

sect	section
align	alignment in bytes
line	virtual line

4.18.3.31 YASM_LIB_DECL void yasm_section_set_default (yasm_section * *sect*, int *def*)

Set section "default" flag to a new value.

Parameters

sect	section
def	new value of default flag

4.18.3.32 YASM_LIB_DECL void yasm_section_set_opt_flags (yasm_section * *sect*, unsigned long *opt_flags*)

Set yasm_optimizer-specific flags.

For yasm_optimizer use only.

Parameters

sect	section
opt_flags	optimizer-specific flags.

4.19 libyasm/symrec.h File Reference

YASM symbol table interface.

Typedefs

- typedef enum yasm_sym_status yasm_sym_status
  Symbol status.
- typedef enum yasm_sym_vis yasm_sym_vis
  Symbol record visibility.
- typedef int(* yasm_symtab_traverse_callback)(yasm_symrec *sym, void *d)
  Callback function for yasm_symrec_traverse().
- typedef struct yasm_symtab_iter yasm_symtab_iter
  Symbol table iterator (opaque type).
- typedef yasm_bytecode * yasm_symrec_get_label_bytecodep
  Dependent pointer to a bytecode.

Enumerations

- enum yasm_sym_status {
  YASM_SYM_NOSTATUS = 0, YASM_SYM_USED = 1 << 0, YASM_SYM_DEFINED = 1 << 1, YASM_SYM_-
  VALUED = 1 << 2,
  YASM_SYM_NOTINTABLE = 1 << 3 }
  Symbol status.

- enum yasm_sym_vis {
  YASM_SYM_LOCAL = 0, YASM_SYM_GLOBAL = 1 << 0, YASM_SYM_COMMON = 1 << 1, YASM_SYM_EXTERN = 1 << 2,
  YASM_SYM_DLOCAL = 1 << 3 }

  Symbol record visibility.

Functions

- YASM_LIB_DECL yasm_symtab * yasm_symtab_create (void)

  Create a new symbol table.
- YASM_LIB_DECL void yasm_symtab_destroy (yasm_symtab *symtab)

  Destroy a symbol table and all internal symbols.
- YASM_LIB_DECL void yasm_symtab_set_case_sensitive (yasm_symtab *symtab, int sensitive)

  Set the symbol table to be case sensitive or not.
- YASM_LIB_DECL yasm_symrec * yasm_symtab_abs_sym (yasm_symtab *symtab)

  Get a reference to the symbol table's "absolute" symbol.
- YASM_LIB_DECL yasm_symrec * yasm_symtab_use (yasm_symtab *symtab, const char *name, unsigned long line)

  Get a reference to (use) a symbol.
- YASM_LIB_DECL yasm_symrec * yasm_symtab_get (yasm_symtab *symtab, const char *name)

  Get a reference to a symbol, without "using" it.
- YASM_LIB_DECL yasm_symrec * yasm_symtab_define_equ (yasm_symtab *symtab, const char *name, yasm_expr *e, unsigned long line)

  Define a symbol as an EQU value.
- YASM_LIB_DECL yasm_symrec * yasm_symtab_define_label (yasm_symtab *symtab, const char *name, yasm_bytecode *precbc, int in_table, unsigned long line)

  Define a symbol as a label.
- YASM_LIB_DECL yasm_symrec * yasm_symtab_define_curpos (yasm_symtab *symtab, const char *name, yasm_bytecode *precbc, unsigned long line)

  Define a symbol as a label representing the current assembly position.
- YASM_LIB_DECL yasm_symrec * yasm_symtab_define_special (yasm_symtab *symtab, const char *name, yasm_sym_vis vis)

  Define a special symbol that will appear in the symbol table and have a defined name, but have no other data associated with it within the standard symrec.
- YASM_LIB_DECL yasm_symrec * yasm_symtab_declare (yasm_symtab *symtab, const char *name, yasm_sym_vis vis, unsigned long line)

  Declare external visibility of a symbol.
- YASM_LIB_DECL void yasm_symrec_declare (yasm_symrec *symrec, yasm_sym_vis vis, unsigned long line)

  Declare external visibility of a symbol.
- YASM_LIB_DECL int yasm_symtab_traverse (yasm_symtab *symtab, void *d, yasm_symtab_traverse_callback func)

  Traverse all symbols in the symbol table.
- YASM_LIB_DECL const yasm_symtab_iter * yasm_symtab_first (const yasm_symtab *symtab)

  Get an iterator pointing to the first symbol in the symbol table.
- YASM_LIB_DECL const yasm_symtab_iter * yasm_symtab_next (const yasm_symtab_iter *prev)

  Move a symbol table iterator to the next symbol in the symbol table.
- YASM_LIB_DECL yasm_symrec * yasm_symtab_iter_value (const yasm_symtab_iter *cur)

Get the symbol corresponding to the current symbol table iterator value.

- YASM_LIB_DECL void yasm_symtab_parser_finalize (yasm_symtab *symtab, int undef_extern, yasm_errwarns *errwarns)

  Finalize symbol table after parsing stage.
- YASM_LIB_DECL void yasm_symtab_print (yasm_symtab *symtab, FILE *f, int indent_level)

  Print the symbol table.
- YASM_LIB_DECL const char * yasm_symrec_get_name (const yasm_symrec *sym)

  Get the name of a symbol.
- YASM_LIB_DECL char * yasm_symrec_get_global_name (const yasm_symrec *sym, const yasm_object *object)

  Get the externally-visible (global) name of a symbol.
- YASM_LIB_DECL yasm_sym_vis yasm_symrec_get_visibility (const yasm_symrec *sym)

  Get the visibility of a symbol.
- YASM_LIB_DECL yasm_sym_status yasm_symrec_get_status (const yasm_symrec *sym)

  Get the status of a symbol.
- YASM_LIB_DECL unsigned long yasm_symrec_get_def_line (const yasm_symrec *sym)

  Get the virtual line of where a symbol was first defined.
- YASM_LIB_DECL unsigned long yasm_symrec_get_decl_line (const yasm_symrec *sym)

  Get the virtual line of where a symbol was first declared.
- YASM_LIB_DECL unsigned long yasm_symrec_get_use_line (const yasm_symrec *sym)

  Get the virtual line of where a symbol was first used.
- YASM_LIB_DECL const yasm_expr * yasm_symrec_get_equ (const yasm_symrec *sym)

  Get EQU value of a symbol.
- YASM_LIB_DECL int yasm_symrec_get_label (const yasm_symrec *sym, yasm_symrec_get_label_bytecodep *precbc)

  Get the label location of a symbol.
- YASM_LIB_DECL void yasm_symrec_set_size (yasm_symrec *sym, int size)

  Set the size of a symbol.
- YASM_LIB_DECL int yasm_symrec_get_size (const yasm_symrec *sym)

  Get the size of a symbol.
- YASM_LIB_DECL void yasm_symrec_set_segment (yasm_symrec *sym, const char *segment)

  Set the segment of a symbol.
- YASM_LIB_DECL const char * yasm_symrec_get_segment (const yasm_symrec *sym)

  Get the segment of a symbol.
- YASM_LIB_DECL int yasm_symrec_is_abs (const yasm_symrec *sym)

  Determine if symbol is the "absolute" symbol created by yasm_symtab_abs_sym().
- YASM_LIB_DECL int yasm_symrec_is_special (const yasm_symrec *sym)

  Determine if symbol is a special symbol.
- YASM_LIB_DECL int yasm_symrec_is_curpos (const yasm_symrec *sym)

  Determine if symbol is a label representing the current assembly position.
- YASM_LIB_DECL void yasm_symrec_set_objext_valparams (yasm_symrec *sym, yasm_valparamhead *objext_valparams)

  Set object-extended valparams.
- YASM_LIB_DECL yasm_valparamhead * yasm_symrec_get_objext_valparams (yasm_symrec *sym)

  Get object-extended valparams, if any, associated with symbol's declaration.
- YASM_LIB_DECL void yasm_symrec_set_common_size (yasm_symrec *sym, yasm_expr *common_size)

  Set common size of symbol.
- YASM_LIB_DECL yasm_expr ** yasm_symrec_get_common_size (yasm_symrec *sym)

Get common size of symbol, if symbol is declared COMMON and a size was set for it.

- YASM_LIB_DECL void * yasm_symrec_get_data (yasm_symrec *sym, const yasm_assoc_data_callback *callback)

  Get associated data for a symbol and data callback.

- YASM_LIB_DECL void yasm_symrec_add_data (yasm_symrec *sym, const yasm_assoc_data_callback *callback, void *data)

  Add associated data to a symbol.

- YASM_LIB_DECL void yasm_symrec_print (const yasm_symrec *sym, FILE *f, int indent_level)

  Print a symbol.

4.19.1 Detailed Description

YASM symbol table interface.

Definition in file symrec.h.

4.19.2 Typedef Documentation

4.19.2.1 typedef enum yasm_sym_status yasm_sym_status

Symbol status.

YASM_SYM_DEFINED is set by yasm_symtab_define_label(), yasm_symtab_define_equ(), or yasm_symtab_declare()/yasm_symrec_declare() with a visibility of YASM_SYM_EXTERN or YASM_SYM_COMMON.

4.19.2.2 typedef enum yasm_sym_vis yasm_sym_vis

Symbol record visibility.

Note

YASM_SYM_EXTERN and YASM_SYM_COMMON are mutually exclusive.

4.19.2.3 typedef yasm_bytecode* yasm_symrec_get_label_bytecodep

Dependent pointer to a bytecode.

Definition at line 313 of file symrec.h.

4.19.2.4 typedef struct yasm_symtab_iter yasm_symtab_iter

Symbol table iterator (opaque type).

Definition at line 210 of file symrec.h.

4.19.2.5 typedef int(* yasm_symtab_traverse_callback)(yasm_symrec *sym,void *d)

Callback function for yasm_symrec_traverse().

Parameters

sym	symbol
d	data passed into yasm_symrec_traverse()

Returns

Nonzero to stop symbol traversal.

Definition at line 195 of file symrec.h.

4.19.3 Enumeration Type Documentation

4.19.3.1 enum yasm_sym_status

Symbol status.

YASM_SYM_DEFINED is set by yasm_symtab_define_label(), yasm_symtab_define_equ(), or yasm_symtab_declare()/yasm_symrec_declare() with a visibility of YASM_SYM_EXTERN or YASM_SYM_COMMON.

Enumerator

YASM_SYM_NOSTATUS no status

YASM_SYM_USED for use before definition

YASM_SYM_DEFINED once it's been defined in the file

YASM_SYM_VALUED once its value has been determined

YASM_SYM_NOTINTABLE if it's not in sym_table (ex. '$')

Definition at line 41 of file symrec.h.

4.19.3.2 enum yasm_sym_vis

Symbol record visibility.

Note

YASM_SYM_EXTERN and YASM_SYM_COMMON are mutually exclusive.

Enumerator

YASM_SYM_LOCAL Default, local only.

YASM_SYM_GLOBAL If symbol is declared GLOBAL.

YASM_SYM_COMMON If symbol is declared COMMON.

YASM_SYM_EXTERN If symbol is declared EXTERN.

YASM_SYM_DLOCAL If symbol is explicitly declared LOCAL.

Definition at line 52 of file symrec.h.

4.19.4 Function Documentation

4.19.4.1 YASM_LIB_DECL void yasm_symrec_add_data (yasm_symrec * *sym*, const yasm_assoc_data_callback * *callback*, void * *data*)

Add associated data to a symbol.

Attention

Deletes any existing associated data for that data callback.

Parameters

sym	symbol
callback	callback
data	data to associate

4.19.4.2 YASM_LIB_DECL void yasm_symrec_declare (yasm_symrec * *symrec,* yasm_sym_vis *vis,* unsigned long *line*)

Declare external visibility of a symbol.

Note

Not all visibility combinations are allowed.

Parameters

symrec	symbol
vis	visibility
line	virtual line of visibility-setting

4.19.4.3 YASM_LIB_DECL yasm_expr** yasm_symrec_get_common_size (yasm_symrec * *sym*)

Get common size of symbol, if symbol is declared COMMON and a size was set for it.

Parameters

sym	symbol

Returns

Common size (NULL if none).

4.19.4.4 YASM_LIB_DECL void* yasm_symrec_get_data (yasm_symrec * *sym,* const yasm_assoc_data_callback * *callback*)

Get associated data for a symbol and data callback.

Parameters

sym	symbol
callback	callback used when adding data

Returns

Associated data (NULL if none).

4.19.4.5 YASM_LIB_DECL unsigned long yasm_symrec_get_decl_line (const yasm_symrec * *sym*)

Get the virtual line of where a symbol was first declared.

Parameters

sym	symbol

Returns

line virtual line

4.19.4.6 YASM_LIB_DECL unsigned long yasm_symrec_get_def_line (const yasm_symrec * *sym*)

Get the virtual line of where a symbol was first defined.

Parameters

sym	symbol

Returns

line virtual line

4.19.4.7 YASM_LIB_DECL const yasm_expr* yasm_symrec_get_equ (const yasm_symrec * *sym*)

Get EQU value of a symbol.

Parameters

sym	symbol

Returns

EQU value, or NULL if symbol is not an EQU or is not defined.

4.19.4.8 YASM_LIB_DECL char* yasm_symrec_get_global_name (const yasm_symrec * *sym,* const yasm_object * *object*)

Get the externally-visible (global) name of a symbol.

Parameters

sym	symbol
object	object

Returns

Externally-visible symbol name (allocated, caller must free).

4.19.4.9 YASM_LIB_DECL int yasm_symrec_get_label (const yasm_symrec * *sym,* yasm_symrec_get_label_bytecodep * *precbc*)

Get the label location of a symbol.

Parameters

sym	symbol
precbc	bytecode preceding label (output)

Returns

0 if not symbol is not a label or if the symbol's visibility is YASM_SYM_EXTERN or YASM_SYM_COMMON (not defined in the file).

4.19.4.10 YASM_LIB_DECL const char* yasm_symrec_get_name (const yasm_symrec * *sym*)

Get the name of a symbol.

Parameters

sym	symbol

Returns

Symbol name.

4.19.4.11 YASM_LIB_DECL yasm_valparamhead* yasm_symrec_get_objext_valparams (yasm_symrec * *sym*)

Get object-extended valparams, if any, associated with symbol's declaration.

Parameters

sym	symbol

Returns

Object-extended valparams (NULL if none).

4.19.4.12 YASM_LIB_DECL const char* yasm_symrec_get_segment (const yasm_symrec * *sym*)

Get the segment of a symbol.

Parameters

sym	symbol

Returns

segment of the symbol, NULL if none specified by the user.

4.19.4.13 YASM_LIB_DECL int yasm_symrec_get_size (const yasm_symrec * *sym*)

Get the size of a symbol.

Parameters

sym	symbol

Returns

size of the symbol, 0 if none specified by the user.

4.19.4.14 YASM_LIB_DECL yasm_sym_status yasm_symrec_get_status (const yasm_symrec * sym)

Get the status of a symbol.

Parameters

sym	symbol

Returns

Symbol status.

4.19.4.15 YASM_LIB_DECL unsigned long yasm_symrec_get_use_line (const yasm_symrec * sym)

Get the virtual line of where a symbol was first used.

Parameters

sym	symbol

Returns

line virtual line

4.19.4.16 YASM_LIB_DECL yasm_sym_vis yasm_symrec_get_visibility (const yasm_symrec * sym)

Get the visibility of a symbol.

Parameters

sym	symbol

Returns

Symbol visibility.

4.19.4.17 YASM_LIB_DECL int yasm_symrec_is_abs (const yasm_symrec * sym)

Determine if symbol is the "absolute" symbol created by yasm_symtab_abs_sym().

Parameters

sym	symbol

Returns

0 if symbol is not the "absolute" symbol, nonzero otherwise.

4.19.4.18 YASM_LIB_DECL int yasm_symrec_is_curpos (const yasm_symrec * sym)

Determine if symbol is a label representing the current assembly position.

Parameters

sym	symbol

Returns

0 if symbol is not a current position label, nonzero otherwise.

4.19.4.19 YASM_LIB_DECL int yasm_symrec_is_special (const yasm_symrec * sym)

Determine if symbol is a special symbol.

Parameters

sym	symbol

Returns

0 if symbol is not a special symbol, nonzero otherwise.

4.19.4.20 YASM_LIB_DECL void yasm_symrec_print (const yasm_symrec * sym, FILE * f, int indent_level)

Print a symbol.

For debugging purposes.

Parameters

f	file
indent_level	indentation level
sym	symbol

4.19.4.21 YASM_LIB_DECL void yasm_symrec_set_common_size (yasm_symrec * sym, yasm_expr * common_size)

Set common size of symbol.

Parameters

sym	symbol
common_size	common size expression

4.19.4.22 YASM_LIB_DECL void yasm_symrec_set_objext_valparams (yasm_symrec * sym, yasm_valparamhead * objext_valparams)

Set object-extended valparams.

Parameters

sym	symbol
objext_valparams	object-extended valparams

4.19.4.23 YASM_LIB_DECL void yasm_symrec_set_segment (yasm_symrec * sym, const char * segment)

Set the segment of a symbol.

Parameters

sym	symbol
segment	segment to be set

4.19.4.24 YASM_LIB_DECL void yasm_symrec_set_size (yasm_symrec * *sym*, int *size*)

Set the size of a symbol.

Parameters

sym	symbol
size	size to be set

4.19.4.25 YASM_LIB_DECL yasm_symrec* yasm_symtab_abs_sym (yasm_symtab * *symtab*)

Get a reference to the symbol table's "absolute" symbol.

This is essentially an EQU with no name and value 0, and is used for relocating absolute current-position-relative values.

See Also

yasm_value_set_curpos_rel().

Parameters

symtab	symbol table

Returns

Absolute symbol (dependent pointer, do not free).

4.19.4.26 YASM_LIB_DECL yasm_symtab* yasm_symtab_create (void)

Create a new symbol table.

4.19.4.27 YASM_LIB_DECL yasm_symrec* yasm_symtab_declare (yasm_symtab * *symtab*, const char * *name*, yasm_sym_vis *vis*, unsigned long *line*)

Declare external visibility of a symbol.

Note

Not all visibility combinations are allowed.

Parameters

symtab	symbol table
name	symbol name
vis	visibility
line	virtual line of visibility-setting

Returns

Symbol (dependent pointer, do not free).

4.19.4.28 YASM_LIB_DECL yasm_symrec* yasm_symtab_define_curpos (yasm_symtab * *symtab*, const char * *name*, yasm_bytecode * *precbc*, unsigned long *line*)

Define a symbol as a label representing the current assembly position.

This should be used for this purpose instead of yasm_symtab_define_label() as value_finalize_scan() looks for usage of this symbol type for special handling. The symbol created is not inserted into the symbol table.

Parameters

symtab	symbol table
name	symbol (label) name
precbc	bytecode preceding label
line	virtual line of label

Returns

Symbol (dependent pointer, do not free).

4.19.4.29 YASM_LIB_DECL yasm_symrec* yasm_symtab_define_equ (yasm_symtab * *symtab*, const char * *name*, yasm_expr * *e*, unsigned long *line*)

Define a symbol as an EQU value.

Parameters

symtab	symbol table
name	symbol (EQU) name
e	EQU value (expression)
line	virtual line of EQU

Returns

Symbol (dependent pointer, do not free).

4.19.4.30 YASM_LIB_DECL yasm_symrec* yasm_symtab_define_label (yasm_symtab * *symtab*, const char * *name*, yasm_bytecode * *precbc*, int *in_table*, unsigned long *line*)

Define a symbol as a label.

Parameters

symtab	symbol table
name	symbol (label) name
precbc	bytecode preceding label
in_table	nonzero if the label should be inserted into the symbol table (some specially-generated ones should not be)
line	virtual line of label

Returns

Symbol (dependent pointer, do not free).

4.19.4.31 YASM_LIB_DECL yasm_symrec* yasm_symtab_define_special (yasm_symtab * *symtab,* const char * *name,* yasm_sym_vis *vis*)

Define a special symbol that will appear in the symbol table and have a defined name, but have no other data associated with it within the standard symrec.

Parameters

symtab	symbol table
name	symbol name
vis	symbol visibility

Returns

Symbol (dependent pointer, do not free).

4.19.4.32 YASM_LIB_DECL void yasm_symtab_destroy (yasm_symtab * *symtab*)

Destroy a symbol table and all internal symbols.

Parameters

symtab	symbol table

Warning

All yasm_symrec *'s into this symbol table become invalid after this is called!

4.19.4.33 YASM_LIB_DECL const yasm_symtab_iter* yasm_symtab_first (const yasm_symtab * *symtab*)

Get an iterator pointing to the first symbol in the symbol table.

Parameters

symtab	symbol table

Returns

Iterator for the symbol table.

4.19.4.34 YASM_LIB_DECL yasm_symrec* yasm_symtab_get (yasm_symtab * *symtab,* const char * *name*)

Get a reference to a symbol, without "using" it.

Should be used for cases when an internal assembler usage of a symbol shouldn't be treated like a normal user symbol usage.

Parameters

symtab	symbol table
name	symbol name

Returns

Symbol (dependent pointer, do not free). May be NULL if symbol doesn't exist.

4.19.4.35 YASM_LIB_DECL yasm_symrec* yasm_symtab_iter_value (const yasm_symtab_iter * *cur*)

Get the symbol corresponding to the current symbol table iterator value.

Parameters

cur	iterator value

Returns

Corresponding symbol.

4.19.4.36 YASM_LIB_DECL const yasm_symtab_iter* yasm_symtab_next (const yasm_symtab_iter * *prev*)

Move a symbol table iterator to the next symbol in the symbol table.

Parameters

prev	Previous iterator value

Returns

Next iterator value, or NULL if no more symbols in the table.

4.19.4.37 YASM_LIB_DECL void yasm_symtab_parser_finalize (yasm_symtab * *symtab,* int *undef_extern,* yasm_errwarns * *errwarns*)

Finalize symbol table after parsing stage.

Checks for symbols that are used but never defined or declared YASM_SYM_EXTERN or YASM_SYM_COMMON.

Parameters

symtab	symbol table
undef_extern	if nonzero, all undef syms should be declared extern
errwarns	error/warning set

Note

Errors/warnings are stored into errwarns.

4.19.4.38 YASM_LIB_DECL void yasm_symtab_print (yasm_symtab * *symtab,* FILE * *f,* int *indent_level*)

Print the symbol table.

For debugging purposes.

Parameters

symtab	symbol table
f	file
indent_level	indentation level

4.19.4.39 YASM_LIB_DECL void yasm_symtab_set_case_sensitive (yasm_symtab * *symtab,* int *sensitive*)

Set the symbol table to be case sensitive or not.

Should be called before adding any symbol.

Parameters

symtab	symbol table
sensitive	whether the symbol table should be case sensitive.

4.19.4.40 YASM_LIB_DECL int yasm_symtab_traverse (yasm_symtab * *symtab,* void * *d,* yasm_symtab_traverse_callback *func*)

Traverse all symbols in the symbol table.

Parameters

symtab	symbol table
d	data to pass to each call of callback function
func	callback function called on each symbol

Returns

Nonzero value returned by callback function if it ever returned nonzero.

4.19.4.41 YASM_LIB_DECL yasm_symrec* yasm_symtab_use (yasm_symtab * *symtab,* const char * *name,* unsigned long *line*)

Get a reference to (use) a symbol.

The symbol does not necessarily need to be defined before it is used.

Parameters

symtab	symbol table
name	symbol name
line	virtual line where referenced

Returns

Symbol (dependent pointer, do not free).

4.20 libyasm/valparam.h File Reference

YASM value/parameter interface.

Data Structures

- struct yasm_valparam

  Value/parameter pair.
- union yasm_valparam::yasm_param

  Parameter value.

- struct yasm_directive

  Directive list entry structure.
- struct yasm_dir_help

  Directive valparam parse helper structure.

Typedefs

- typedef struct yasm_dir_help yasm_dir_help

  Directive valparam parse helper structure.

Functions

- STAILQ_HEAD (yasm_valparamhead, yasm_valparam)

  Linked list of value/parameter pairs.
- YASM_LIB_DECL void yasm_call_directive (const yasm_directive *directive, yasm_object *object, yasm_valparamhead *valparams, yasm_valparamhead *objext_valparams, unsigned long line)

  Call a directive.
- YASM_LIB_DECL yasm_valparam * yasm_vp_create_id (char *v, char *p, int id_prefix)

  Create a new valparam with identifier parameter.
- YASM_LIB_DECL yasm_valparam * yasm_vp_create_string (char *v, char *p)

  Create a new valparam with string parameter.
- YASM_LIB_DECL yasm_valparam * yasm_vp_create_expr (char *v, yasm_expr *p)

  Create a new valparam with expression parameter.
- YASM_LIB_DECL yasm_expr * yasm_vp_expr (const yasm_valparam *vp, yasm_symtab *symtab, unsigned long line)

  Get a valparam parameter as an expr.
- YASM_LIB_DECL const char * yasm_vp_string (const yasm_valparam *vp)

  Get a valparam parameter as a string.
- YASM_LIB_DECL const char * yasm_vp_id (const yasm_valparam *vp)

  Get a valparam parameter as an identifier.
- YASM_LIB_DECL yasm_valparamhead * yasm_vps_create (void)

  Create a new linked list of valparams.
- YASM_LIB_DECL void yasm_vps_destroy (yasm_valparamhead *headp)

  Destroy a list of valparams (created with yasm_vps_create).
- void yasm_vps_initialize (yasm_valparamhead *headp)

  Initialize linked list of valparams.
- YASM_LIB_DECL void yasm_vps_delete (yasm_valparamhead *headp)

  Destroy (free allocated memory for) linked list of valparams (created with yasm_vps_initialize).
- void yasm_vps_append (yasm_valparamhead *headp, yasm_valparam *vp)

  Append valparam to tail of linked list.
- yasm_valparam * yasm_vps_first (yasm_valparamhead *headp)

  Get first valparam in linked list.
- yasm_valparam * yasm_vps_next (yasm_valparam *cur)

  Get next valparam in linked list.
- YASM_LIB_DECL void yasm_vps_print (const yasm_valparamhead *headp, FILE *f)

  Iterate through linked list of valparams.

- YASM_LIB_DECL int yasm_dir_helper (void *obj, yasm_valparam *vp_first, unsigned long line, const yasm_dir_help *help, size_t nhelp, void *data, int(*helper_valparam)(void *object, yasm_valparam *vp, unsigned long line, void *data))

  Help parse a list of directive value/parameters.
- YASM_LIB_DECL int yasm_dir_helper_flag_set (void *obj, yasm_valparam *vp, unsigned long line, void *data, uintptr_t arg)

  Standard helper for yasm_dir_helper() that simply sets a flag when called.
- YASM_LIB_DECL int yasm_dir_helper_flag_or (void *obj, yasm_valparam *vp, unsigned long line, void *data, uintptr_t arg)

  Standard helper for yasm_dir_helper() that simply ORs a flag when called.
- YASM_LIB_DECL int yasm_dir_helper_flag_and (void *obj, yasm_valparam *vp, unsigned long line, void *data, uintptr_t arg)

  Standard helper for yasm_dir_helper() that simply ANDs a flag when called.
- YASM_LIB_DECL int yasm_dir_helper_expr (void *obj, yasm_valparam *vp, unsigned long line, void *data, uintptr_t arg)

  Standard helper for yasm_dir_helper() that parses an expr parameter.
- YASM_LIB_DECL int yasm_dir_helper_intn (void *obj, yasm_valparam *vp, unsigned long line, void *data, uintptr_t arg)

  Standard helper for yasm_dir_helper() that parses an intnum parameter.
- YASM_LIB_DECL int yasm_dir_helper_string (void *obj, yasm_valparam *vp, unsigned long line, void *data, uintptr_t arg)

  Standard helper for yasm_dir_helper() that parses an string (or standalone identifier) parameter.
- YASM_LIB_DECL int yasm_dir_helper_valparam_warn (void *obj, yasm_valparam *vp, unsigned long line, void *data)

  Standard catch-all callback fro yasm_dir_helper().

4.20.1 Detailed Description

YASM value/parameter interface.

Definition in file valparam.h.

4.20.2 Typedef Documentation

4.20.2.1 typedef struct yasm_dir_help yasm_dir_help

Directive valparam parse helper structure.

4.20.3 Function Documentation

4.20.3.1 STAILQ_HEAD (yasm_valparamhead , yasm_valparam)

Linked list of value/parameter pairs.

4.20.3.2 YASM_LIB_DECL void yasm_call_directive (const yasm_directive * *directive*, yasm_object * *object*, yasm_valparamhead * *valparams*, yasm_valparamhead * *object_valparams*, unsigned long *line*)

Call a directive.

Performs any valparam checks asked for by the directive prior to call. Note that for a variety of reasons, a directive can generate an error.

Parameters

directive	directive
object	object
valparams	value/parameters
objext_valparams	object format-specific value/parameters
line	virtual line (from yasm_linemap)

4.20.3.3 YASM_LIB_DECL int yasm_dir_helper (void * *obj*, yasm_valparam * *vp_first*, unsigned long *line*, const yasm_dir_help * *help*, size_t *nhelp*, void * *data*, int(*)(void *object, yasm_valparam *vp, unsigned long line, void *data) *helper_valparam*)

Help parse a list of directive value/parameters.

Takes an array of yasm_dir_help structures and tries to match val=param (or just val) against the passed value/parameters. When no match is found in the array of help structures, calls helper_valparam.

Parameters

obj	object to be passed to yasm_dir_help.helper() or helper_valparam() callback
vp_first	first value/parameter to examine
line	virtual line number; passed down to helper callback
help	array of yasm_dir_help structures
nhelp	number of array elements
data	base data pointer; if a match is found, the respective yasm_dir_help.off is added to this prior to it being passed to the helper callback
helper_valparam	catch-all callback; should return -1 on error, 0 if not matched, 1 if matched.

Returns

-1 on error, 1 if any arguments matched (including via catch-all callback), 0 if no match.

4.20.3.4 YASM_LIB_DECL int yasm_dir_helper_expr (void * *obj*, yasm_valparam * *vp*, unsigned long *line*, void * *data*, uintptr_t *arg*)

Standard helper for yasm_dir_helper() that parses an expr parameter.

The yasm_dir_help structure that uses this function should have needsparam=1. The obj parameter to yasm_dir_helper() when this helper is used MUST point to a yasm_object. In addition, the data parameter that is ultimately passed to this function (e.g. yasm_dir_helper() data parameter plus yasm_dir_help.off) must point to a yasm_expr * initialized to NULL.

Parameters

obj	object; must be yasm_object
vp	valparam
line	virtual line number
data	pointer to yasm_expr *
arg	unused argument

Returns

-1 on error, 0 otherwise.

4.20.3.5 YASM_LIB_DECL int yasm_dir_helper_flag_and (void * *obj*, yasm_valparam * *vp*, unsigned long *line*, void * *data*, uintptr_t *arg*)

Standard helper for yasm_dir_helper() that simply ANDs a flag when called.

It does not look at the vp; rather, it uses the value of the arg parameter, and ANDs its inverse (~) with the unsigned long value in data.

Parameters

obj	unused
vp	unused
line	unused
data	pointer to an unsigned long
arg	flag to AND

Returns

0

4.20.3.6 YASM_LIB_DECL int yasm_dir_helper_flag_or (void * *obj*, yasm_valparam * *vp*, unsigned long *line*, void * *data*, uintptr_t *arg*)

Standard helper for yasm_dir_helper() that simply ORs a flag when called.

It does not look at the vp; rather, it uses the value of the arg parameter, and ORs it with the unsigned long value in data.

Parameters

obj	unused
vp	unused
line	unused
data	pointer to an unsigned long
arg	flag to OR

Returns

0

4.20.3.7 YASM_LIB_DECL int yasm_dir_helper_flag_set (void * *obj*, yasm_valparam * *vp*, unsigned long *line*, void * *data*, uintptr_t *arg*)

Standard helper for yasm_dir_helper() that simply sets a flag when called.

It does not look at the vp; rather, it uses the value of the arg parameter, and stores an unsigned long value to data.

Parameters

obj	unused
vp	unused
line	unused
data	pointer to an unsigned long
arg	flag to set

Returns

0

4.20.3.8 YASM_LIB_DECL int yasm_dir_helper_intn (void * *obj*, yasm_valparam * *vp*, unsigned long *line*, void * *data*, uintptr_t *arg*)

Standard helper for yasm_dir_helper() that parses an intnum parameter.

The yasm_dir_help structure that uses this function should have needsparam=1. The obj parameter to yasm_dir_helper() when this helper is used MUST point to a yasm_object. In addition, the data parameter that is ultimately passed to this function (e.g. yasm_dir_helper() data parameter plus yasm_dir_help.off) must point to a yasm_intnum * initialized to NULL.

Parameters

obj	object; must be yasm_object
vp	valparam
line	virtual line number
data	pointer to yasm_intnum *
arg	unused argument

Returns

-1 on error, 0 otherwise.

4.20.3.9 YASM_LIB_DECL int yasm_dir_helper_string (void * *obj*, yasm_valparam * *vp*, unsigned long *line*, void * *data*, uintptr_t *arg*)

Standard helper for yasm_dir_helper() that parses an string (or standalone identifier) parameter.

The yasm_dir_help structure that uses this function should have needsparam=1. The data parameter that is ultimately passed to this function (e.g. yasm_dir_helper() data parameter plus yasm_dir_help.off) must point to a char * initialized to NULL.

Parameters

obj	unused
vp	valparam
line	unused
data	pointer to char *
arg	unused

Returns

-1 on error, 0 otherwise.

4.20.3.10 YASM_LIB_DECL int yasm_dir_helper_valparam_warn (void * *obj*, yasm_valparam * *vp*, unsigned long *line*, void * *data*)

Standard catch-all callback fro yasm_dir_helper().

Generates standard warning for all valparams.

Parameters

obj	unused
vp	valparam
line	unused
data	unused

Returns

0

4.20.3.11 YASM_LIB_DECL yasm_valparam* yasm_vp_create_expr (char * v, yasm_expr * p)

Create a new valparam with expression parameter.

Parameters

v	value
p	parameter

Returns

Newly allocated valparam.

4.20.3.12 YASM_LIB_DECL yasm_valparam* yasm_vp_create_id (char * v, char * p, int id_prefix)

Create a new valparam with identifier parameter.

Parameters

v	value
p	parameter
id_prefix	identifier prefix for raw identifiers

Returns

Newly allocated valparam.

4.20.3.13 YASM_LIB_DECL yasm_valparam* yasm_vp_create_string (char * v, char * p)

Create a new valparam with string parameter.

Parameters

v	value
p	parameter

Returns

Newly allocated valparam.

4.20.3.14 YASM_LIB_DECL yasm_expr* yasm_vp_expr (const yasm_valparam * *vp*, yasm_symtab * *symtab*, unsigned long *line*)

Get a valparam parameter as an expr.

If the parameter is an identifier, it's treated as a symbol (yasm_symtab_use() is called to convert it).

Parameters

vp	valparam
symtab	symbol table
line	virtual line

Returns

Expression, or NULL if vp is NULL or the parameter cannot be converted to an expression.

4.20.3.15 YASM_LIB_DECL const char* yasm_vp_id (const yasm_valparam * *vp*)

Get a valparam parameter as an identifier.

Parameters

vp	valparam

Returns

Identifier (string), or NULL if vp is NULL or the parameter is not an identifier.

4.20.3.16 YASM_LIB_DECL const char* yasm_vp_string (const yasm_valparam * *vp*)

Get a valparam parameter as a string.

If the parameter is an identifier, it's treated as a string.

Parameters

vp	valparam

Returns

String, or NULL if vp is NULL or the parameter cannot be realized as a string.

4.20.3.17 void yasm_vps_append (yasm_valparamhead * *headp*, yasm_valparam * *vp*)

Append valparam to tail of linked list.

Parameters

headp	linked list
vp	valparam

4.20.3.18 YASM_LIB_DECL yasm_valparamhead* yasm_vps_create (void)

Create a new linked list of valparams.

Returns

Newly allocated valparam list.

4.20.3.19 YASM_LIB_DECL void yasm_vps_delete (yasm_valparamhead * *headp*)

Destroy (free allocated memory for) linked list of valparams (created with yasm_vps_initialize).

Warning

Deletes val/params.

Parameters

headp	linked list

4.20.3.20 YASM_LIB_DECL void yasm_vps_destroy (yasm_valparamhead * *headp*)

Destroy a list of valparams (created with yasm_vps_create).

Parameters

headp	list of valparams

4.20.3.21 yasm_valparam* yasm_vps_first (yasm_valparamhead * *headp*)

Get first valparam in linked list.

Parameters

headp	linked list

Returns

First valparam in linked list.

4.20.3.22 void yasm_vps_initialize (yasm_valparamhead * *headp*)

Initialize linked list of valparams.

Parameters

headp	linked list

4.20.3.23 yasm_valparam* yasm_vps_next (yasm_valparam * *cur*)

Get next valparam in linked list.

Parameters

cur	previous valparam in linked list

Returns

Next valparam in linked list.

4.20.3.24 YASM_LIB_DECL void yasm_vps_print (const yasm_valparamhead * *headp*, FILE * *f*)

Iterate through linked list of valparams.

Parameters

iter	iterator variable
headp	linked list Print linked list of valparams. For debugging purposes.
f	file
headp	linked list

4.21 libyasm/value.h File Reference

YASM value interface.

Functions

- YASM_LIB_DECL void yasm_value_initialize (yasm_value *value, yasm_expr *e, unsigned int size)

  Initialize a yasm_value with just an expression.
- YASM_LIB_DECL void yasm_value_init_sym (yasm_value *value, yasm_symrec *sym, unsigned int size)

  Initialize a yasm_value with just a symrec.
- YASM_LIB_DECL void yasm_value_init_copy (yasm_value *value, const yasm_value *orig)

  Initialize a yasm_value as a copy of another yasm_value.
- YASM_LIB_DECL void yasm_value_delete (yasm_value *value)

  Frees any memory inside value; does not free value itself.
- YASM_LIB_DECL void yasm_value_set_curpos_rel (yasm_value *value, yasm_bytecode *bc, unsigned int ip_rel)

  Set a value to be relative to the current assembly position rather than relative to the section start.
- YASM_LIB_DECL int yasm_value_finalize (yasm_value *value, yasm_bytecode *precbc)

  Perform yasm_value_finalize_expr() on a value that already exists from being initialized with yasm_value_initialize().
- YASM_LIB_DECL int yasm_value_finalize_expr (yasm_value *value, yasm_expr *e, yasm_bytecode *precbc, unsigned int size)

  Break a yasm_expr into a yasm_value constituent parts.
- YASM_LIB_DECL yasm_intnum * yasm_value_get_intnum (yasm_value *value, yasm_bytecode *bc, int calc_bc_dist)

  Get value if absolute or PC-relative section-local relative.
- YASM_LIB_DECL int yasm_value_output_basic (yasm_value *value, unsigned char *buf, size_t destsize, yasm_bytecode *bc, int warn, yasm_arch *arch)

  Output value if constant or PC-relative section-local.
- YASM_LIB_DECL void yasm_value_print (const yasm_value *value, FILE *f, int indent_level)

  Print a value.

4.21.1 Detailed Description

YASM value interface.

Definition in file value.h.

4.21.2 Function Documentation

4.21.2.1 YASM_LIB_DECL void yasm_value_delete (yasm_value * value)

Frees any memory inside value; does not free value itself.

Parameters

value	value

4.21.2.2 YASM_LIB_DECL int yasm_value_finalize (yasm_value * value, yasm_bytecode * precbc)

Perform yasm_value_finalize_expr() on a value that already exists from being initialized with yasm_value_initialize().

Parameters

value	value
precbc	previous bytecode to bytecode containing value

Returns

Nonzero if value could not be split.

4.21.2.3 YASM_LIB_DECL int yasm_value_finalize_expr (yasm_value * value, yasm_expr * e, yasm_bytecode * precbc, unsigned int size)

Break a yasm_expr into a yasm_value constituent parts.

Extracts the relative portion of the value, SEG and WRT portions, and top-level right shift, if any. Places the remaining expr into the absolute portion of the value. Essentially a combination of yasm_value_initialize() and yasm_value_finalize(). First expands references to symrecs in absolute sections by expanding with the absolute section start plus the symrec offset within the absolute section.

Parameters

value	value to store split portions into
e	expression input
precbc	previous bytecode to bytecode containing expression
size	value size (in bits)

Returns

Nonzero if the expr could not be split into a value for some reason (e.g. the relative portion was not added, but multiplied, etc).

Warning

Do not use e after this call. Even if an error is returned, e is stored into value.

Note

This should only be called after the parse is complete. Calling before the parse is complete will usually result in an error return.

4.21.2.4 YASM_LIB_DECL yasm_intnum* yasm_value_get_intnum (yasm_value * *value*, yasm_bytecode * *bc*, int *calc_bc_dist*)

Get value if absolute or PC-relative section-local relative.

Returns NULL otherwise.

Parameters

value	value
bc	current bytecode (for PC-relative calculation); if NULL, NULL is returned for PC-relative values.
calc_bc_dist	if nonzero, calculates bytecode distances in absolute portion of value

Note

Adds in value.rel (correctly) if PC-relative and in the same section as bc (and there is no WRT or SEG).

Returns

Intnum if can be resolved to integer value, otherwise NULL.

4.21.2.5 YASM_LIB_DECL void yasm_value_init_copy (yasm_value * *value*, const yasm_value * *orig*)

Initialize a yasm_value as a copy of another yasm_value.

Any expressions within orig are copied, so it's safe to delete the copy.

Parameters

value	value (copy to create)
orig	original value

4.21.2.6 YASM_LIB_DECL void yasm_value_init_sym (yasm_value * *value*, yasm_symrec * *sym*, unsigned int *size*)

Initialize a yasm_value with just a symrec.

No processing is performed, the symrec is simply stuck into value.rel and the other fields are initialized.

Parameters

value	value to be initialized
sym	symrec
size	value size (in bits)

4.21.2.7 YASM_LIB_DECL void yasm_value_initialize (yasm_value * *value*, yasm_expr * *e*, unsigned int *size*)

Initialize a yasm_value with just an expression.

No processing is performed, the expression is simply stuck into value.abs and the other fields are initialized. Use yasm_expr_extract_value() to perform "smart" processing into a yasm_value. This function is intended for use during parsing simply to ensure all fields of the value are initialized; after the parse is complete, yasm_value_extract() should be called

to finalize the value. The value defaults to unsigned.

Parameters

value	value to be initialized
e	expression (kept)
size	value size (in bits)

4.21.2.8 YASM_LIB_DECL int yasm_value_output_basic (yasm_value * *value*, unsigned char * *buf*, size_t *destsize*, yasm_bytecode * *bc*, int *warn*, yasm_arch * *arch*)

Output value if constant or PC-relative section-local.

This should be used from objfmt yasm_output_value_func() functions. functions.

Parameters

value	value
buf	buffer for byte representation
destsize	destination size (in bytes)
bc	current bytecode (usually passed into higher-level calling function)
warn	enables standard warnings: zero for none; nonzero for overflow/underflow floating point and integer warnings
arch	architecture

Note

Adds in value.rel (correctly) if PC-relative and in the same section as bc (and there is no WRT or SEG); if this is not the desired behavior, e.g. a reloc is needed in this case, don't use this function!

Returns

0 if no value output due to value needing relocation; 1 if value output; -1 if error.

4.21.2.9 YASM_LIB_DECL void yasm_value_print (const yasm_value * *value*, FILE * *f*, int *indent_level*)

Print a value.

For debugging purposes.

Parameters

value	value
indent_level	indentation level
f	file

4.21.2.10 YASM_LIB_DECL void yasm_value_set_curpos_rel (yasm_value * *value*, yasm_bytecode * *bc*, unsigned int *ip_rel*)

Set a value to be relative to the current assembly position rather than relative to the section start.

Parameters

value	value
bc	bytecode containing value
ip_rel	if nonzero, indicates IP-relative data relocation, sometimes used to generate special relocations

Note

If value is just an absolute value, will get an absolute symrec to reference to (via bc's symbol table).

Index

abs
  yasm_value, 36
add_default_section
  yasm_objfmt_module, 25
add_include_file
  yasm_preproc_module, 30
add_standard
  yasm_preproc_module, 30
arch.h
  YASM_ARCH_CREATE_BAD_MACHINE, 41
  YASM_ARCH_CREATE_BAD_PARSER, 41
  YASM_ARCH_CREATE_OK, 41
  YASM_ARCH_INSN, 41
  YASM_ARCH_NOTINSNPREFIX, 41
  YASM_ARCH_NOTREGTMOD, 41
  YASM_ARCH_PREFIX, 41
  YASM_ARCH_REG, 41
  YASM_ARCH_REGGROUP, 42
  YASM_ARCH_SEGREG, 42
  YASM_ARCH_TARGETMOD, 42
arch.h
  yasm_arch_create, 42
  yasm_arch_create_empty_insn, 42
  yasm_arch_create_error, 40, 41
  yasm_arch_destroy, 42
  yasm_arch_ea_create, 42
  yasm_arch_ea_destroy, 43
  yasm_arch_ea_print, 43
  yasm_arch_floatnum_tobytes, 43
  yasm_arch_get_address_size, 44
  yasm_arch_get_fill, 44
  yasm_arch_get_machine, 44
  yasm_arch_get_reg_size, 44
  yasm_arch_insnprefix, 40, 41
  yasm_arch_intnum_tobytes, 44
  yasm_arch_keyword, 45
  yasm_arch_machine, 41
  yasm_arch_min_insn_len, 45
  yasm_arch_module, 41
  yasm_arch_name, 45
  yasm_arch_parse_check_insnprefix, 46
  yasm_arch_parse_check_regtmod, 46
  yasm_arch_reg_print, 46
  yasm_arch_reggroup_get_reg, 47
  yasm_arch_regtmod, 41
  yasm_arch_segreg_print, 47
  yasm_arch_set_var, 47
  yasm_arch_wordsize, 47
assocdat.h
  yasm__assoc_data, 48
  yasm__assoc_data_add, 48
  yasm__assoc_data_create, 49
  yasm__assoc_data_destroy, 49
  yasm__assoc_data_get, 49
  yasm__assoc_data_print, 49

bc_index
  yasm_bytecode, 11
bytecode.h
  STAILQ_HEAD, 53
  yasm_bc__add_symrec, 53
  yasm_bc__next, 52
  yasm_bc_add_span_func, 52
  yasm_bc_calc_len, 53
  yasm_bc_calc_len_common, 54
  yasm_bc_create_align, 54
  yasm_bc_create_common, 54
  yasm_bc_create_data, 54
  yasm_bc_create_incbin, 55
  yasm_bc_create_leb128, 55
  yasm_bc_create_org, 55
  yasm_bc_create_reserve, 56
  yasm_bc_destroy, 56
  yasm_bc_elem_size, 56
  yasm_bc_expand, 56
  yasm_bc_expand_common, 57
  yasm_bc_finalize, 57
  yasm_bc_get_insn, 57
  yasm_bc_get_multiple, 57
  yasm_bc_get_multiple_expr, 58
  yasm_bc_get_section, 58
  yasm_bc_next_offset, 58
  yasm_bc_print, 58
  yasm_bc_reserve_numitems, 59
  yasm_bc_set_multiple, 59
  yasm_bc_tobytes, 59
  yasm_bc_tobytes_common, 60
  yasm_bc_transform, 60
  yasm_bytecode_callback, 53
  yasm_calc_bc_dist, 60
  yasm_dataval, 53
  yasm_datavalhead, 53
  yasm_dv_create_expr, 60
  yasm_dv_create_raw, 60
  yasm_dv_create_reserve, 61
  yasm_dv_create_string, 61
  yasm_dv_get_multiple, 61
  yasm_dv_get_value, 61
  yasm_dv_set_multiple, 62
  yasm_dvs_append, 62
  yasm_dvs_delete, 62
  yasm_dvs_initialize, 62

yasm_dvs_print, 62

calc_len
yasm_bytecode_callback, 13
callback
yasm_bytecode, 11
contents
yasm_bytecode, 11
coretype.h
YASM_EXPR_ADD, 71
YASM_EXPR_AND, 71
YASM_EXPR_DIV, 71
YASM_EXPR_EQ, 72
YASM_EXPR_GE, 72
YASM_EXPR_GT, 72
YASM_EXPR_IDENT, 71
YASM_EXPR_LAND, 71
YASM_EXPR_LE, 72
YASM_EXPR_LNOR, 72
YASM_EXPR_LNOT, 71
YASM_EXPR_LOGIC, 71
YASM_EXPR_LOR, 71
YASM_EXPR_LT, 72
YASM_EXPR_LXNOR, 72
YASM_EXPR_LXOR, 71
YASM_EXPR_MOD, 71
YASM_EXPR_MUL, 71
YASM_EXPR_NE, 72
YASM_EXPR_NEG, 71
YASM_EXPR_NONNUM, 72
YASM_EXPR_NOR, 71
YASM_EXPR_NOT, 71
YASM_EXPR_OR, 71
YASM_EXPR_SEG, 72
YASM_EXPR_SEGOFF, 72
YASM_EXPR_SHL, 71
YASM_EXPR_SHR, 71
YASM_EXPR_SIGNDIV, 71
YASM_EXPR_SIGNMOD, 71
YASM_EXPR_SUB, 71
YASM_EXPR_WRT, 72
YASM_EXPR_XNOR, 71
YASM_EXPR_XOR, 71
coretype.h
yasm__mergesort, 72
yasm__strcasecmp, 72
yasm__strncasecmp, 72
yasm__strsep, 73
yasm__xstrdup, 73
yasm__xstrndup, 73
yasm_arch, 66
yasm_assoc_data_callback, 66
yasm_bytecode, 66
yasm_dbgfmt, 66
yasm_dbgfmt_module, 66
yasm_directive, 66
yasm_effaddr, 66
yasm_errwarns, 67
yasm_expr, 67
yasm_expr_op, 67, 71
yasm_floatnum, 67
yasm_insn, 67
yasm_intnum, 67
yasm_linemap, 68
yasm_listfmt, 68
yasm_object, 68
yasm_objfmt, 68
yasm_objfmt_module, 68
yasm_output_reloc_func, 68
yasm_output_value_func, 69
yasm_parser, 69
yasm_preproc, 70
yasm_section, 70
yasm_stdmac, 70
yasm_symrec, 70
yasm_symtab, 70
yasm_valparam, 70
yasm_valparamhead, 70
yasm_value, 71
yasm_xcalloc, 73
yasm_xfree, 74
yasm_xmalloc, 74
yasm_xrealloc, 74
create
yasm_arch_module, 6
yasm_dbgfmt_module, 16
yasm_listfmt_module, 22
yasm_objfmt_module, 25
yasm_preproc_module, 30
create_empty_insn
yasm_arch_module, 6
cur_section
yasm_object, 24
curpos_rel
yasm_value, 36

dbgfmt.h
yasm_dbgfmt_create, 75
yasm_dbgfmt_destroy, 75
yasm_dbgfmt_generate, 75
yasm_dbgfmt_keyword, 76
dbgfmt_keywords
yasm_objfmt_module, 26
default_machine_keyword
yasm_arch_module, 6
default_preproc_keyword
yasm_parser_module, 29
default_x86_mode_bits

yasm_objfmt_module, 26
define_builtin
yasm_preproc_module, 31
destroy
yasm_arch_module, 6
yasm_assoc_data_callback, 9
yasm_bytecode_callback, 14
yasm_dbgfmt_module, 17
yasm_listfmt_module, 22
yasm_objfmt_module, 26
yasm_preproc_module, 31
directives
yasm_arch_module, 6
yasm_dbgfmt_module, 17
yasm_objfmt_module, 26
do_parse
yasm_parser_module, 29

ea_create
yasm_arch_module, 6
ea_destroy
yasm_arch_module, 6
ea_print
yasm_arch_module, 6
elem_size
yasm_bytecode_callback, 14
errwarn.h
YASM_ERROR_ARITHMETIC, 79
YASM_ERROR_ASSERTION, 79
YASM_ERROR_FLOATING_POINT, 79
YASM_ERROR_GENERAL, 79
YASM_ERROR_IO, 80
YASM_ERROR_NONE, 79
YASM_ERROR_NOT_ABSOLUTE, 79
YASM_ERROR_NOT_CONSTANT, 80
YASM_ERROR_NOT_IMPLEMENTED, 80
YASM_ERROR_OVERFLOW, 79
YASM_ERROR_PARSE, 80
YASM_ERROR_SYNTAX, 80
YASM_ERROR_TOO_COMPLEX, 79
YASM_ERROR_TYPE, 80
YASM_ERROR_VALUE, 79
YASM_ERROR_ZERO_DIVISION, 79
YASM_WARN_GENERAL, 80
YASM_WARN_IMPLICIT_SIZE_OVERRIDE, 80
YASM_WARN_NONE, 80
YASM_WARN_ORPHAN_LABEL, 80
YASM_WARN_PREPROC, 80
YASM_WARN_SIZE_OVERRIDE, 80
YASM_WARN_UNINIT_CONTENTS, 80
YASM_WARN_UNREC_CHAR, 80
errwarn.h
yasm__conv_unprint, 80
yasm__fatal, 80
yasm_error_class, 78, 79
yasm_error_clear, 81
yasm_error_fetch, 81
yasm_error_matches, 81
yasm_error_occurred, 81
yasm_error_set, 81
yasm_error_set_va, 82
yasm_error_set_xref, 82
yasm_error_set_xref_va, 82
yasm_errwarn_initialize, 82
yasm_errwarn_propagate, 82
yasm_errwarns_create, 83
yasm_errwarns_destroy, 83
yasm_errwarns_num_errors, 83
yasm_errwarns_output_all, 83
yasm_fatal, 85
yasm_gettext_hook, 85
yasm_internal_error, 78
yasm_internal_error_, 85
yasm_print_error_func, 79
yasm_print_warning_func, 79
yasm_warn_class, 79, 80
yasm_warn_clear, 83
yasm_warn_disable, 83
yasm_warn_disable_all, 84
yasm_warn_enable, 84
yasm_warn_fetch, 84
yasm_warn_occurred, 84
yasm_warn_set, 84
yasm_warn_set_va, 84
expand
yasm_bytecode_callback, 14
expr.h
YASM_EXPR_EXPR, 90
YASM_EXPR_FLOAT, 90
YASM_EXPR_INT, 90
YASM_EXPR_NONE, 90
YASM_EXPR_PRECBC, 90
YASM_EXPR_REG, 90
YASM_EXPR_SUBST, 90
YASM_EXPR_SYM, 90
expr.h
yasm_expr__bc_dist_subst, 90
yasm_expr__contains, 90
yasm_expr__copy_except, 91
yasm_expr__item, 89
yasm_expr__level_tree, 91
yasm_expr__order_terms, 91
yasm_expr__subst, 92
yasm_expr__traverse_leaves_in, 92
yasm_expr__traverse_leaves_in_const, 92
yasm_expr__type, 89, 90
yasm_expr_copy, 93
yasm_expr_create, 93

yasm_expr_create_branch, 88
yasm_expr_create_ident, 88
yasm_expr_create_tree, 89
yasm_expr_destroy, 93
yasm_expr_expr, 93
yasm_expr_extract_deep_segoff, 94
yasm_expr_extract_segoff, 94
yasm_expr_extract_wrt, 94
yasm_expr_float, 94
yasm_expr_get_intnum, 95
yasm_expr_get_reg, 95
yasm_expr_get_symrec, 95
yasm_expr_int, 95
yasm_expr_is_op, 96
yasm_expr_precbc, 96
yasm_expr_print, 96
yasm_expr_reg, 96
yasm_expr_segment, 96
yasm_expr_simplify, 89
yasm_expr_size, 97
yasm_expr_sym, 97
yasm_expr_xform_func, 89
extension
yasm_objfmt_module, 26

file.h
YASM_LOAD_16_B, 100
YASM_LOAD_16_L, 101
YASM_LOAD_32_B, 101
YASM_LOAD_32_L, 102
YASM_LOAD_8, 102
YASM_READ_16_B, 102
YASM_READ_16_L, 103
YASM_READ_32_B, 103
YASM_READ_32_L, 103
YASM_READ_8, 104
YASM_SAVE_16_B, 104
YASM_SAVE_16_L, 105
YASM_SAVE_32_B, 105
YASM_SAVE_32_L, 105
YASM_SAVE_8, 106
YASM_WRITE_16_B, 106
YASM_WRITE_16_L, 107
YASM_WRITE_32_B, 107
YASM_WRITE_32_L, 107
YASM_WRITE_8, 108
yasm__abspath, 108
yasm__combpath, 99
yasm__combpath_unix, 108
yasm__combpath_win, 109
yasm__createpath, 100
yasm__createpath_common, 109
yasm__getcwd, 109
yasm__splitpath, 100
yasm__splitpath_unix, 109
yasm__splitpath_win, 110
yasm_add_include_path, 110
yasm_fill_helper, 110
yasm_fopen_include, 110
yasm_fwrite_16_b, 111
yasm_fwrite_16_l, 111
yasm_fwrite_32_b, 111
yasm_fwrite_32_l, 112
yasm_scanner, 108
yasm_scanner_delete, 112
yasm_scanner_initialize, 112
yasm_unescape_cstring, 112
finalize
yasm_bytecode_callback, 15
floatnum.h
yasm_floatnum_calc, 113
yasm_floatnum_check_size, 114
yasm_floatnum_cleanup, 114
yasm_floatnum_copy, 114
yasm_floatnum_create, 114
yasm_floatnum_destroy, 114
yasm_floatnum_get_int, 115
yasm_floatnum_get_sized, 115
yasm_floatnum_initialize, 115
yasm_floatnum_print, 115
floatnum_tobytes
yasm_arch_module, 7

generate
yasm_dbgfmt_module, 17
get_address_size
yasm_arch_module, 7
get_fill
yasm_arch_module, 7
get_included_file
yasm_preproc_module, 31
get_line
yasm_preproc_module, 31
get_machine
yasm_arch_module, 7
get_reg_size
yasm_arch_module, 7
get_special_sym
yasm_objfmt_module, 26

HAMT
hamt.h, 117
HAMT_create
hamt.h, 117
HAMT_destroy
hamt.h, 117
HAMT_first
hamt.h, 117
HAMT_insert

hamt.h, 117
HAMT_next
hamt.h, 118
HAMT_search
hamt.h, 118
HAMT_traverse
hamt.h, 118
HAMTEntry
hamt.h, 117
HAMTEntry_get_data
hamt.h, 119
hamt.h
HAMT, 117
HAMT_create, 117
HAMT_destroy, 117
HAMT_first, 117
HAMT_insert, 117
HAMT_next, 118
HAMT_search, 118
HAMT_traverse, 118
HAMTEntry, 117
HAMTEntry_get_data, 119
handler
yasm_directive, 19
helper
yasm_dir_help, 18

id_at_ok
yasm_objfmt_module, 26
id_prefix
yasm_valparam, 35
init_new_section
yasm_objfmt_module, 26
intnum.h
yasm_get_sleb128, 121
yasm_get_uleb128, 121
yasm_intnum_calc, 121
yasm_intnum_check_size, 122
yasm_intnum_cleanup, 122
yasm_intnum_compare, 122
yasm_intnum_copy, 122
yasm_intnum_create_bin, 122
yasm_intnum_create_charconst_nasm, 123
yasm_intnum_create_charconst_tasm, 123
yasm_intnum_create_dec, 123
yasm_intnum_create_hex, 123
yasm_intnum_create_int, 124
yasm_intnum_create_leb128, 124
yasm_intnum_create_oct, 124
yasm_intnum_create_sized, 124
yasm_intnum_create_uint, 125
yasm_intnum_destroy, 125
yasm_intnum_get_int, 125
yasm_intnum_get_leb128, 125
yasm_intnum_get_sized, 126
yasm_intnum_get_str, 126
yasm_intnum_get_uint, 126
yasm_intnum_in_range, 126
yasm_intnum_initialize, 127
yasm_intnum_is_neg1, 127
yasm_intnum_is_pos1, 127
yasm_intnum_is_zero, 127
yasm_intnum_print, 127
yasm_intnum_set, 128
yasm_intnum_set_int, 128
yasm_intnum_set_uint, 128
yasm_intnum_sign, 128
yasm_intnum_size_leb128, 128
yasm_intnum_zero, 129
yasm_size_sleb128, 129
yasm_size_uleb128, 129
intnum_tobytes
yasm_arch_module, 7
ip_rel
yasm_value, 36

jump_target
yasm_value, 36

keyword
yasm_arch_machine, 4
yasm_arch_module, 7
yasm_dbgfmt_module, 17
yasm_listfmt_module, 22
yasm_objfmt_module, 27
yasm_preproc_module, 31

len
yasm_bytecode, 11
libyasm.h, 38
libyasm/arch.h, 38
libyasm/assocdat.h, 48
libyasm/bytecode.h, 49
libyasm/coretype.h, 63
libyasm/dbgfmt.h, 74
libyasm/errwarn.h, 76
libyasm/expr.h, 86
libyasm/file.h, 97
libyasm/floatnum.h, 113
libyasm/hamt.h, 116
libyasm/intnum.h, 119
libyasm/linemap.h, 129
libyasm/listfmt.h, 133
libyasm/objfmt.h, 135
libyasm/parser.h, 137
libyasm/preproc.h, 138
libyasm/section.h, 141
libyasm/symrec.h, 151
libyasm/valparam.h, 165

libyasm/value.h, 174
line
    yasm_bytecode, 11
    yasm_expr, 20
linemap.h
    yasm_linemap_add_source, 130
    yasm_linemap_create, 131
    yasm_linemap_destroy, 131
    yasm_linemap_get_current, 131
    yasm_linemap_get_source, 131
    yasm_linemap_goto_next, 131
    yasm_linemap_lookup, 132
    yasm_linemap_poke, 132
    yasm_linemap_set, 132
    yasm_linemap_traverse_filenames, 133
listfmt.h
    yasm_listfmt_create, 134
    yasm_listfmt_destroy, 134
    yasm_listfmt_keyword, 134
    yasm_listfmt_module, 134
    yasm_listfmt_output, 134

machines
    yasm_arch_module, 7
macros
    yasm_stdmac, 34
min_insn_len
    yasm_arch_module, 8
mult_int
    yasm_bytecode, 11
multiple
    yasm_bytecode, 11

name
    yasm_arch_machine, 4
    yasm_arch_module, 8
    yasm_dbgfmt_module, 17
    yasm_directive, 20
    yasm_listfmt_module, 23
    yasm_objfmt_module, 27
    yasm_preproc_module, 31
needsparam
    yasm_dir_help, 18
no_warn
    yasm_value, 37
numterms
    yasm_expr, 20

objfmt.h
    yasm_objfmt_add_default_section, 135
    yasm_objfmt_create, 136
    yasm_objfmt_destroy, 136
    yasm_objfmt_get_special_sym, 136
    yasm_objfmt_init_new_section, 136
    yasm_objfmt_output, 136
    yasm_objfmt_section_switch, 137
off
    yasm_dir_help, 18
offset
    yasm_bytecode, 12
op
    yasm_expr, 20
output
    yasm_listfmt_module, 23
    yasm_objfmt_module, 27

parse_check_insnprefix
    yasm_arch_module, 8
parse_check_regtmod
    yasm_arch_module, 8
parser.h
    yasm_parser_module, 138
predefine_macro
    yasm_preproc_module, 32
preproc.h
    yasm_preproc_add_include_file, 139
    yasm_preproc_add_standard, 139
    yasm_preproc_create, 139
    yasm_preproc_define_builtin, 139
    yasm_preproc_destroy, 140
    yasm_preproc_get_included_file, 140
    yasm_preproc_get_line, 140
    yasm_preproc_module, 139
    yasm_preproc_predefine_macro, 140
    yasm_preproc_undefine_macro, 140
preproc_keywords
    yasm_parser_module, 29
print
    yasm_assoc_data_callback, 9
    yasm_bytecode_callback, 15

reg_print
    yasm_arch_module, 8
reggroup_get_reg
    yasm_arch_module, 8
rel
    yasm_value, 37
rshift
    yasm_value, 37

STAILQ_ENTRY
    yasm_bytecode, 11
STAILQ_HEAD
    bytecode.h, 53
    valparam.h, 167
    yasm_object, 24
section.h
    yasm_object_create, 143
    yasm_object_destroy, 143
    yasm_object_directive, 144

yasm_object_finalize, 144
yasm_object_find_general, 144
yasm_object_get_general, 144
yasm_object_optimize, 145
yasm_object_print, 145
yasm_object_sections_traverse, 145
yasm_object_set_global_prefix, 146
yasm_object_set_global_suffix, 146
yasm_object_set_source_fn, 146
yasm_reloc, 143
yasm_reloc_get, 146
yasm_section_add_data, 146
yasm_section_add_reloc, 147
yasm_section_bcs_append, 147
yasm_section_bcs_first, 147
yasm_section_bcs_last, 147
yasm_section_bcs_traverse, 148
yasm_section_get_align, 148
yasm_section_get_data, 148
yasm_section_get_name, 149
yasm_section_get_object, 149
yasm_section_get_opt_flags, 149
yasm_section_is_code, 149
yasm_section_is_default, 149
yasm_section_print, 150
yasm_section_reloc_next, 150
yasm_section_relocs_first, 150
yasm_section_set_align, 150
yasm_section_set_default, 151
yasm_section_set_opt_flags, 151
section_rel
yasm_value, 37
section_switch
yasm_objfmt_module, 27
seg_of
yasm_value, 37
segreg_print
yasm_arch_module, 8
set_var
yasm_arch_module, 8
sign
yasm_value, 37
size
yasm_value, 37
stdmacs
yasm_objfmt_module, 27
yasm_parser_module, 29
symrec.h
YASM_SYM_COMMON, 155
YASM_SYM_DEFINED, 155
YASM_SYM_DLOCAL, 155
YASM_SYM_EXTERN, 155
YASM_SYM_GLOBAL, 155
YASM_SYM_LOCAL, 155
YASM_SYM_NOSTATUS, 155
YASM_SYM_NOTINTABLE, 155
YASM_SYM_USED, 155
YASM_SYM_VALUED, 155
symrec.h
yasm_sym_status, 154, 155
yasm_sym_vis, 154, 155
yasm_symrec_add_data, 155
yasm_symrec_declare, 156
yasm_symrec_get_common_size, 156
yasm_symrec_get_data, 156
yasm_symrec_get_decl_line, 156
yasm_symrec_get_def_line, 157
yasm_symrec_get_equ, 157
yasm_symrec_get_global_name, 157
yasm_symrec_get_label, 157
yasm_symrec_get_label_bytecodep, 154
yasm_symrec_get_name, 158
yasm_symrec_get_objext_valparams, 158
yasm_symrec_get_segment, 158
yasm_symrec_get_size, 158
yasm_symrec_get_status, 158
yasm_symrec_get_use_line, 159
yasm_symrec_get_visibility, 159
yasm_symrec_is_abs, 159
yasm_symrec_is_curpos, 159
yasm_symrec_is_special, 160
yasm_symrec_print, 160
yasm_symrec_set_common_size, 160
yasm_symrec_set_objext_valparams, 160
yasm_symrec_set_segment, 160
yasm_symrec_set_size, 161
yasm_symtab_abs_sym, 161
yasm_symtab_create, 161
yasm_symtab_declare, 161
yasm_symtab_define_curpos, 162
yasm_symtab_define_equ, 162
yasm_symtab_define_label, 162
yasm_symtab_define_special, 163
yasm_symtab_destroy, 163
yasm_symtab_first, 163
yasm_symtab_get, 163
yasm_symtab_iter, 154
yasm_symtab_iter_value, 164
yasm_symtab_next, 164
yasm_symtab_parser_finalize, 164
yasm_symtab_print, 164
yasm_symtab_set_case_sensitive, 165
yasm_symtab_traverse, 165
yasm_symtab_traverse_callback, 154
yasm_symtab_use, 165
symrecs
yasm_bytecode, 12

terms
yasm_expr, 21
tobytes
yasm_bytecode_callback, 15

undefine_macro
yasm_preproc_module, 32

valparam.h
STAILQ_HEAD, 167
yasm_call_directive, 167
yasm_dir_help, 167
yasm_dir_helper, 168
yasm_dir_helper_expr, 168
yasm_dir_helper_flag_and, 169
yasm_dir_helper_flag_or, 169
yasm_dir_helper_flag_set, 169
yasm_dir_helper_intn, 170
yasm_dir_helper_string, 170
yasm_dir_helper_valparam_warn, 170
yasm_vp_create_expr, 171
yasm_vp_create_id, 171
yasm_vp_create_string, 171
yasm_vp_expr, 172
yasm_vp_id, 172
yasm_vp_string, 172
yasm_vps_append, 172
yasm_vps_create, 173
yasm_vps_delete, 173
yasm_vps_destroy, 173
yasm_vps_first, 173
yasm_vps_initialize, 173
yasm_vps_next, 173
yasm_vps_print, 174
value.h
yasm_value_delete, 175
yasm_value_finalize, 175
yasm_value_finalize_expr, 175
yasm_value_get_intnum, 176
yasm_value_init_copy, 176
yasm_value_init_sym, 176
yasm_value_initialize, 176
yasm_value_output_basic, 177
yasm_value_print, 177
yasm_value_set_curpos_rel, 177

wordsize
yasm_arch_module, 9
wrt
yasm_value, 37

YASM_ARCH_CREATE_BAD_MACHINE
arch.h, 41
YASM_ARCH_CREATE_BAD_PARSER
arch.h, 41
YASM_ARCH_CREATE_OK
arch.h, 41
YASM_ARCH_INSN
arch.h, 41
YASM_ARCH_NOTINSNPREFIX
arch.h, 41
YASM_ARCH_NOTREGTMOD
arch.h, 41
YASM_ARCH_PREFIX
arch.h, 41
YASM_ARCH_REG
arch.h, 41
YASM_ARCH_REGGROUP
arch.h, 42
YASM_ARCH_SEGREG
arch.h, 42
YASM_ARCH_TARGETMOD
arch.h, 42
YASM_BC_SPECIAL_INSN
yasm_bytecode_callback, 13
YASM_BC_SPECIAL_OFFSET
yasm_bytecode_callback, 13
YASM_BC_SPECIAL_RESERVE
yasm_bytecode_callback, 13
YASM_DIR_ANY
yasm_directive, 19
YASM_DIR_ARG_REQUIRED
yasm_directive, 19
YASM_DIR_ID_REQUIRED
yasm_directive, 19
YASM_ERROR_ARITHMETIC
errwarn.h, 79
YASM_ERROR_ASSERTION
errwarn.h, 79
YASM_ERROR_FLOATING_POINT
errwarn.h, 79
YASM_ERROR_GENERAL
errwarn.h, 79
YASM_ERROR_IO
errwarn.h, 80
YASM_ERROR_NONE
errwarn.h, 79
YASM_ERROR_NOT_ABSOLUTE
errwarn.h, 79
YASM_ERROR_NOT_CONSTANT
errwarn.h, 80
YASM_ERROR_NOT_IMPLEMENTED
errwarn.h, 80
YASM_ERROR_OVERFLOW
errwarn.h, 79
YASM_ERROR_PARSE
errwarn.h, 80
YASM_ERROR_SYNTAX
errwarn.h, 80

YASM_ERROR_TOO_COMPLEX
errwarn.h, 79
YASM_ERROR_TYPE
errwarn.h, 80
YASM_ERROR_VALUE
errwarn.h, 79
YASM_ERROR_ZERO_DIVISION
errwarn.h, 79
YASM_EXPR_ADD
coretype.h, 71
YASM_EXPR_AND
coretype.h, 71
YASM_EXPR_DIV
coretype.h, 71
YASM_EXPR_EQ
coretype.h, 72
YASM_EXPR_EXPR
expr.h, 90
YASM_EXPR_FLOAT
expr.h, 90
YASM_EXPR_GE
coretype.h, 72
YASM_EXPR_GT
coretype.h, 72
YASM_EXPR_IDENT
coretype.h, 71
YASM_EXPR_INT
expr.h, 90
YASM_EXPR_LAND
coretype.h, 71
YASM_EXPR_LE
coretype.h, 72
YASM_EXPR_LNOR
coretype.h, 72
YASM_EXPR_LNOT
coretype.h, 71
YASM_EXPR_LOGIC
coretype.h, 71
YASM_EXPR_LOR
coretype.h, 71
YASM_EXPR_LT
coretype.h, 72
YASM_EXPR_LXNOR
coretype.h, 72
YASM_EXPR_LXOR
coretype.h, 71
YASM_EXPR_MOD
coretype.h, 71
YASM_EXPR_MUL
coretype.h, 71
YASM_EXPR_NE
coretype.h, 72
YASM_EXPR_NEG
coretype.h, 71
YASM_EXPR_NONE
expr.h, 90
YASM_EXPR_NONNUM
coretype.h, 72
YASM_EXPR_NOR
coretype.h, 71
YASM_EXPR_NOT
coretype.h, 71
YASM_EXPR_OR
coretype.h, 71
YASM_EXPR_PRECBC
expr.h, 90
YASM_EXPR_REG
expr.h, 90
YASM_EXPR_SEG
coretype.h, 72
YASM_EXPR_SEGOFF
coretype.h, 72
YASM_EXPR_SHL
coretype.h, 71
YASM_EXPR_SHR
coretype.h, 71
YASM_EXPR_SIGNDIV
coretype.h, 71
YASM_EXPR_SIGNMOD
coretype.h, 71
YASM_EXPR_SUB
coretype.h, 71
YASM_EXPR_SUBST
expr.h, 90
YASM_EXPR_SYM
expr.h, 90
YASM_EXPR_WRT
coretype.h, 72
YASM_EXPR_XNOR
coretype.h, 71
YASM_EXPR_XOR
coretype.h, 71
YASM_PARAM_EXPR
yasm_valparam, 35
YASM_PARAM_ID
yasm_valparam, 35
YASM_PARAM_STRING
yasm_valparam, 35
YASM_SYM_COMMON
symrec.h, 155
YASM_SYM_DEFINED
symrec.h, 155
YASM_SYM_DLOCAL
symrec.h, 155
YASM_SYM_EXTERN
symrec.h, 155
YASM_SYM_GLOBAL
symrec.h, 155

YASM_SYM_LOCAL
symrec.h, 155
YASM_SYM_NOSTATUS
symrec.h, 155
YASM_SYM_NOTINTABLE
symrec.h, 155
YASM_SYM_USED
symrec.h, 155
YASM_SYM_VALUED
symrec.h, 155
YASM_WARN_GENERAL
errwarn.h, 80
YASM_WARN_IMPLICIT_SIZE_OVERRIDE
errwarn.h, 80
YASM_WARN_NONE
errwarn.h, 80
YASM_WARN_ORPHAN_LABEL
errwarn.h, 80
YASM_WARN_PREPROC
errwarn.h, 80
YASM_WARN_SIZE_OVERRIDE
errwarn.h, 80
YASM_WARN_UNINIT_CONTENTS
errwarn.h, 80
YASM_WARN_UNREC_CHAR
errwarn.h, 80
YASM_LOAD_16_B
file.h, 100
YASM_LOAD_16_L
file.h, 101
YASM_LOAD_32_B
file.h, 101
YASM_LOAD_32_L
file.h, 102
YASM_LOAD_8
file.h, 102
YASM_READ_16_B
file.h, 102
YASM_READ_16_L
file.h, 103
YASM_READ_32_B
file.h, 103
YASM_READ_32_L
file.h, 103
YASM_READ_8
file.h, 104
YASM_SAVE_16_B
file.h, 104
YASM_SAVE_16_L
file.h, 105
YASM_SAVE_32_B
file.h, 105
YASM_SAVE_32_L
file.h, 105
YASM_SAVE_8
file.h, 106
YASM_WRITE_16_B
file.h, 106
YASM_WRITE_16_L
file.h, 107
YASM_WRITE_32_B
file.h, 107
YASM_WRITE_32_L
file.h, 107
YASM_WRITE_8
file.h, 108
yasm_bytecode_callback
YASM_BC_SPECIAL_INSN, 13
YASM_BC_SPECIAL_OFFSET, 13
YASM_BC_SPECIAL_RESERVE, 13
yasm_directive
YASM_DIR_ANY, 19
YASM_DIR_ARG_REQUIRED, 19
YASM_DIR_ID_REQUIRED, 19
yasm_valparam
YASM_PARAM_EXPR, 35
YASM_PARAM_ID, 35
YASM_PARAM_STRING, 35
yasm__abspath
file.h, 108
yasm__assoc_data
assocdat.h, 48
yasm__assoc_data_add
assocdat.h, 48
yasm__assoc_data_create
assocdat.h, 49
yasm__assoc_data_destroy
assocdat.h, 49
yasm__assoc_data_get
assocdat.h, 49
yasm__assoc_data_print
assocdat.h, 49
yasm__combpath
file.h, 99
yasm__combpath_unix
file.h, 108
yasm__combpath_win
file.h, 109
yasm__conv_unprint
errwarn.h, 80
yasm__createpath
file.h, 100
yasm__createpath_common
file.h, 109
yasm__fatal
errwarn.h, 80
yasm__getcwd
file.h, 109

yasm__mergesort
coretype.h, 72
yasm__splitpath
file.h, 100
yasm__splitpath_unix
file.h, 109
yasm__splitpath_win
file.h, 110
yasm__strcasecmp
coretype.h, 72
yasm__strncasecmp
coretype.h, 72
yasm__strsep
coretype.h, 73
yasm__xstrdup
coretype.h, 73
yasm__xstrndup
coretype.h, 73
yasm_add_include_path
file.h, 110
yasm_arch
coretype.h, 66
yasm_arch_create
arch.h, 42
yasm_arch_create_empty_insn
arch.h, 42
yasm_arch_create_error
arch.h, 40, 41
yasm_arch_destroy
arch.h, 42
yasm_arch_ea_create
arch.h, 42
yasm_arch_ea_destroy
arch.h, 43
yasm_arch_ea_print
arch.h, 43
yasm_arch_floatnum_tobytes
arch.h, 43
yasm_arch_get_address_size
arch.h, 44
yasm_arch_get_fill
arch.h, 44
yasm_arch_get_machine
arch.h, 44
yasm_arch_get_reg_size
arch.h, 44
yasm_arch_insnprefix
arch.h, 40, 41
yasm_arch_intnum_tobytes
arch.h, 44
yasm_arch_keyword
arch.h, 45
yasm_arch_machine, 3
arch.h, 41
keyword, 4
name, 4
yasm_arch_min_insn_len
arch.h, 45
yasm_arch_module, 4
arch.h, 41
create, 6
create_empty_insn, 6
default_machine_keyword, 6
destroy, 6
directives, 6
ea_create, 6
ea_destroy, 6
ea_print, 6
floatnum_tobytes, 7
get_address_size, 7
get_fill, 7
get_machine, 7
get_reg_size, 7
intnum_tobytes, 7
keyword, 7
machines, 7
min_insn_len, 8
name, 8
parse_check_insnprefix, 8
parse_check_regtmod, 8
reg_print, 8
reggroup_get_reg, 8
segreg_print, 8
set_var, 8
wordsize, 9
yasm_arch_name
arch.h, 45
yasm_arch_parse_check_insnprefix
arch.h, 46
yasm_arch_parse_check_regtmod
arch.h, 46
yasm_arch_reg_print
arch.h, 46
yasm_arch_reggroup_get_reg
arch.h, 47
yasm_arch_regtmod
arch.h, 41
yasm_arch_segreg_print
arch.h, 47
yasm_arch_set_var
arch.h, 47
yasm_arch_wordsize
arch.h, 47
yasm_assoc_data_callback, 9
coretype.h, 66
destroy, 9
print, 9
yasm_bc__add_symrec

bytecode.h, 53
yasm_bc__next
bytecode.h, 52
yasm_bc_add_span_func
bytecode.h, 52
yasm_bc_calc_len
bytecode.h, 53
yasm_bc_calc_len_common
bytecode.h, 54
yasm_bc_create_align
bytecode.h, 54
yasm_bc_create_common
bytecode.h, 54
yasm_bc_create_data
bytecode.h, 54
yasm_bc_create_incbin
bytecode.h, 55
yasm_bc_create_leb128
bytecode.h, 55
yasm_bc_create_org
bytecode.h, 55
yasm_bc_create_reserve
bytecode.h, 56
yasm_bc_destroy
bytecode.h, 56
yasm_bc_elem_size
bytecode.h, 56
yasm_bc_expand
bytecode.h, 56
yasm_bc_expand_common
bytecode.h, 57
yasm_bc_finalize
bytecode.h, 57
yasm_bc_get_insn
bytecode.h, 57
yasm_bc_get_multiple
bytecode.h, 57
yasm_bc_get_multiple_expr
bytecode.h, 58
yasm_bc_get_section
bytecode.h, 58
yasm_bc_next_offset
bytecode.h, 58
yasm_bc_print
bytecode.h, 58
yasm_bc_reserve_numitems
bytecode.h, 59
yasm_bc_set_multiple
bytecode.h, 59
yasm_bc_tobytes
bytecode.h, 59
yasm_bc_tobytes_common
bytecode.h, 60
yasm_bc_transform
bytecode.h, 60
yasm_bytecode, 10
bc_index, 11
callback, 11
contents, 11
coretype.h, 66
len, 11
line, 11
mult_int, 11
multiple, 11
offset, 12
STAILQ_ENTRY, 11
symrecs, 12
yasm_bytecode_callback, 12
bytecode.h, 53
calc_len, 13
destroy, 14
elem_size, 14
expand, 14
finalize, 15
print, 15
tobytes, 15
yasm_bytecode_special_type, 13
yasm_bytecode_special_type
yasm_bytecode_callback, 13
yasm_calc_bc_dist
bytecode.h, 60
yasm_call_directive
valparam.h, 167
yasm_dataval
bytecode.h, 53
yasm_datavalhead
bytecode.h, 53
yasm_dbgfmt
coretype.h, 66
yasm_dbgfmt_create
dbgfmt.h, 75
yasm_dbgfmt_destroy
dbgfmt.h, 75
yasm_dbgfmt_generate
dbgfmt.h, 75
yasm_dbgfmt_keyword
dbgfmt.h, 76
yasm_dbgfmt_module, 16
coretype.h, 66
create, 16
destroy, 17
directives, 17
generate, 17
keyword, 17
name, 17
yasm_dir_help, 17
helper, 18
needsparam, 18

off, 18
valparam.h, 167
yasm_dir_helper
valparam.h, 168
yasm_dir_helper_expr
valparam.h, 168
yasm_dir_helper_flag_and
valparam.h, 169
yasm_dir_helper_flag_or
valparam.h, 169
yasm_dir_helper_flag_set
valparam.h, 169
yasm_dir_helper_intn
valparam.h, 170
yasm_dir_helper_string
valparam.h, 170
yasm_dir_helper_valparam_warn
valparam.h, 170
yasm_directive, 18
coretype.h, 66
handler, 19
name, 20
yasm_directive_flags, 19
yasm_directive_flags
yasm_directive, 19
yasm_dv_create_expr
bytecode.h, 60
yasm_dv_create_raw
bytecode.h, 60
yasm_dv_create_reserve
bytecode.h, 61
yasm_dv_create_string
bytecode.h, 61
yasm_dv_get_multiple
bytecode.h, 61
yasm_dv_get_value
bytecode.h, 61
yasm_dv_set_multiple
bytecode.h, 62
yasm_dvs_append
bytecode.h, 62
yasm_dvs_delete
bytecode.h, 62
yasm_dvs_initialize
bytecode.h, 62
yasm_dvs_print
bytecode.h, 62
yasm_effaddr
coretype.h, 66
yasm_error_class
errwarn.h, 78, 79
yasm_error_clear
errwarn.h, 81
yasm_error_fetch
errwarn.h, 81
yasm_error_matches
errwarn.h, 81
yasm_error_occurred
errwarn.h, 81
yasm_error_set
errwarn.h, 81
yasm_error_set_va
errwarn.h, 82
yasm_error_set_xref
errwarn.h, 82
yasm_error_set_xref_va
errwarn.h, 82
yasm_errwarn_initialize
errwarn.h, 82
yasm_errwarn_propagate
errwarn.h, 82
yasm_errwarns
coretype.h, 67
yasm_errwarns_create
errwarn.h, 83
yasm_errwarns_destroy
errwarn.h, 83
yasm_errwarns_num_errors
errwarn.h, 83
yasm_errwarns_output_all
errwarn.h, 83
yasm_expr, 20
coretype.h, 67
line, 20
numterms, 20
op, 20
terms, 21
yasm_expr__bc_dist_subst
expr.h, 90
yasm_expr__contains
expr.h, 90
yasm_expr__copy_except
expr.h, 91
yasm_expr__item, 21
expr.h, 89
yasm_expr__level_tree
expr.h, 91
yasm_expr__order_terms
expr.h, 91
yasm_expr__subst
expr.h, 92
yasm_expr__traverse_leaves_in
expr.h, 92
yasm_expr__traverse_leaves_in_const
expr.h, 92
yasm_expr__type
expr.h, 89, 90
yasm_expr_copy

expr.h, 93
yasm_expr_create
expr.h, 93
yasm_expr_create_branch
expr.h, 88
yasm_expr_create_ident
expr.h, 88
yasm_expr_create_tree
expr.h, 89
yasm_expr_destroy
expr.h, 93
yasm_expr_expr
expr.h, 93
yasm_expr_extract_deep_segoff
expr.h, 94
yasm_expr_extract_segoff
expr.h, 94
yasm_expr_extract_wrt
expr.h, 94
yasm_expr_float
expr.h, 94
yasm_expr_get_intnum
expr.h, 95
yasm_expr_get_reg
expr.h, 95
yasm_expr_get_symrec
expr.h, 95
yasm_expr_int
expr.h, 95
yasm_expr_is_op
expr.h, 96
yasm_expr_op
coretype.h, 67, 71
yasm_expr_precbc
expr.h, 96
yasm_expr_print
expr.h, 96
yasm_expr_reg
expr.h, 96
yasm_expr_segment
expr.h, 96
yasm_expr_simplify
expr.h, 89
yasm_expr_size
expr.h, 97
yasm_expr_sym
expr.h, 97
yasm_expr_xform_func
expr.h, 89
yasm_fatal
errwarn.h, 85
yasm_fill_helper
file.h, 110
yasm_floatnum
coretype.h, 67
yasm_floatnum_calc
floatnum.h, 113
yasm_floatnum_check_size
floatnum.h, 114
yasm_floatnum_cleanup
floatnum.h, 114
yasm_floatnum_copy
floatnum.h, 114
yasm_floatnum_create
floatnum.h, 114
yasm_floatnum_destroy
floatnum.h, 114
yasm_floatnum_get_int
floatnum.h, 115
yasm_floatnum_get_sized
floatnum.h, 115
yasm_floatnum_initialize
floatnum.h, 115
yasm_floatnum_print
floatnum.h, 115
yasm_fopen_include
file.h, 110
yasm_fwrite_16_b
file.h, 111
yasm_fwrite_16_l
file.h, 111
yasm_fwrite_32_b
file.h, 111
yasm_fwrite_32_l
file.h, 112
yasm_get_sleb128
intnum.h, 121
yasm_get_uleb128
intnum.h, 121
yasm_gettext_hook
errwarn.h, 85
yasm_insn
coretype.h, 67
yasm_internal_error
errwarn.h, 78
yasm_internal_error_
errwarn.h, 85
yasm_intnum
coretype.h, 67
yasm_intnum_calc
intnum.h, 121
yasm_intnum_check_size
intnum.h, 122
yasm_intnum_cleanup
intnum.h, 122
yasm_intnum_compare
intnum.h, 122
yasm_intnum_copy

intnum.h, 122
yasm_intnum_create_bin
intnum.h, 122
yasm_intnum_create_charconst_nasm
intnum.h, 123
yasm_intnum_create_charconst_tasm
intnum.h, 123
yasm_intnum_create_dec
intnum.h, 123
yasm_intnum_create_hex
intnum.h, 123
yasm_intnum_create_int
intnum.h, 124
yasm_intnum_create_leb128
intnum.h, 124
yasm_intnum_create_oct
intnum.h, 124
yasm_intnum_create_sized
intnum.h, 124
yasm_intnum_create_uint
intnum.h, 125
yasm_intnum_destroy
intnum.h, 125
yasm_intnum_get_int
intnum.h, 125
yasm_intnum_get_leb128
intnum.h, 125
yasm_intnum_get_sized
intnum.h, 126
yasm_intnum_get_str
intnum.h, 126
yasm_intnum_get_uint
intnum.h, 126
yasm_intnum_in_range
intnum.h, 126
yasm_intnum_initialize
intnum.h, 127
yasm_intnum_is_neg1
intnum.h, 127
yasm_intnum_is_pos1
intnum.h, 127
yasm_intnum_is_zero
intnum.h, 127
yasm_intnum_print
intnum.h, 127
yasm_intnum_set
intnum.h, 128
yasm_intnum_set_int
intnum.h, 128
yasm_intnum_set_uint
intnum.h, 128
yasm_intnum_sign
intnum.h, 128
yasm_intnum_size_leb128
intnum.h, 128
yasm_intnum_zero
intnum.h, 129
yasm_linemap
coretype.h, 68
yasm_linemap_add_source
linemap.h, 130
yasm_linemap_create
linemap.h, 131
yasm_linemap_destroy
linemap.h, 131
yasm_linemap_get_current
linemap.h, 131
yasm_linemap_get_source
linemap.h, 131
yasm_linemap_goto_next
linemap.h, 131
yasm_linemap_lookup
linemap.h, 132
yasm_linemap_poke
linemap.h, 132
yasm_linemap_set
linemap.h, 132
yasm_linemap_traverse_filenames
linemap.h, 133
yasm_listfmt
coretype.h, 68
yasm_listfmt_create
listfmt.h, 134
yasm_listfmt_destroy
listfmt.h, 134
yasm_listfmt_keyword
listfmt.h, 134
yasm_listfmt_module, 22
create, 22
destroy, 22
keyword, 22
listfmt.h, 134
name, 23
output, 23
yasm_listfmt_output
listfmt.h, 134
yasm_object, 23
coretype.h, 68
cur_section, 24
STAILQ_HEAD, 24
yasm_object_create
section.h, 143
yasm_object_destroy
section.h, 143
yasm_object_directive
section.h, 144
yasm_object_finalize
section.h, 144

yasm_object_find_general
section.h, 144
yasm_object_get_general
section.h, 144
yasm_object_optimize
section.h, 145
yasm_object_print
section.h, 145
yasm_object_sections_traverse
section.h, 145
yasm_object_set_global_prefix
section.h, 146
yasm_object_set_global_suffix
section.h, 146
yasm_object_set_source_fn
section.h, 146
yasm_objfmt
coretype.h, 68
yasm_objfmt_add_default_section
objfmt.h, 135
yasm_objfmt_create
objfmt.h, 136
yasm_objfmt_destroy
objfmt.h, 136
yasm_objfmt_get_special_sym
objfmt.h, 136
yasm_objfmt_init_new_section
objfmt.h, 136
yasm_objfmt_module, 24
add_default_section, 25
coretype.h, 68
create, 25
dbgfmt_keywords, 26
default_x86_mode_bits, 26
destroy, 26
directives, 26
extension, 26
get_special_sym, 26
id_at_ok, 26
init_new_section, 26
keyword, 27
name, 27
output, 27
section_switch, 27
stdmacs, 27
yasm_objfmt_output
objfmt.h, 136
yasm_objfmt_section_switch
objfmt.h, 137
yasm_output_reloc_func
coretype.h, 68
yasm_output_value_func
coretype.h, 69
yasm_overrides, 27
yasm_param_type
yasm_valparam, 35
yasm_parser
coretype.h, 69
yasm_parser_module, 28
default_preproc_keyword, 29
do_parse, 29
parser.h, 138
preproc_keywords, 29
stdmacs, 29
yasm_preproc
coretype.h, 70
yasm_preproc_add_include_file
preproc.h, 139
yasm_preproc_add_standard
preproc.h, 139
yasm_preproc_create
preproc.h, 139
yasm_preproc_define_builtin
preproc.h, 139
yasm_preproc_destroy
preproc.h, 140
yasm_preproc_get_included_file
preproc.h, 140
yasm_preproc_get_line
preproc.h, 140
yasm_preproc_module, 29
add_include_file, 30
add_standard, 30
create, 30
define_builtin, 31
destroy, 31
get_included_file, 31
get_line, 31
keyword, 31
name, 31
predefine_macro, 32
preproc.h, 139
undefine_macro, 32
yasm_preproc_predefine_macro
preproc.h, 140
yasm_preproc_undefine_macro
preproc.h, 140
yasm_print_error_func
errwarn.h, 79
yasm_print_warning_func
errwarn.h, 79
yasm_reloc, 32
section.h, 143
yasm_reloc_get
section.h, 146
yasm_scanner, 32
file.h, 108
yasm_scanner_delete

file.h, 112
yasm_scanner_initialize
file.h, 112
yasm_section
coretype.h, 70
yasm_section_add_data
section.h, 146
yasm_section_add_reloc
section.h, 147
yasm_section_bcs_append
section.h, 147
yasm_section_bcs_first
section.h, 147
yasm_section_bcs_last
section.h, 147
yasm_section_bcs_traverse
section.h, 148
yasm_section_get_align
section.h, 148
yasm_section_get_data
section.h, 148
yasm_section_get_name
section.h, 149
yasm_section_get_object
section.h, 149
yasm_section_get_opt_flags
section.h, 149
yasm_section_is_code
section.h, 149
yasm_section_is_default
section.h, 149
yasm_section_print
section.h, 150
yasm_section_reloc_next
section.h, 150
yasm_section_relocs_first
section.h, 150
yasm_section_set_align
section.h, 150
yasm_section_set_default
section.h, 151
yasm_section_set_opt_flags
section.h, 151
yasm_size_sleb128
intnum.h, 129
yasm_size_uleb128
intnum.h, 129
yasm_stdmac, 33
coretype.h, 70
macros, 34
yasm_sym_status
symrec.h, 154, 155
yasm_sym_vis
symrec.h, 154, 155
yasm_symrec
coretype.h, 70
yasm_symrec_add_data
symrec.h, 155
yasm_symrec_declare
symrec.h, 156
yasm_symrec_get_common_size
symrec.h, 156
yasm_symrec_get_data
symrec.h, 156
yasm_symrec_get_decl_line
symrec.h, 156
yasm_symrec_get_def_line
symrec.h, 157
yasm_symrec_get_equ
symrec.h, 157
yasm_symrec_get_global_name
symrec.h, 157
yasm_symrec_get_label
symrec.h, 157
yasm_symrec_get_label_bytecodep
symrec.h, 154
yasm_symrec_get_name
symrec.h, 158
yasm_symrec_get_objext_valparams
symrec.h, 158
yasm_symrec_get_segment
symrec.h, 158
yasm_symrec_get_size
symrec.h, 158
yasm_symrec_get_status
symrec.h, 158
yasm_symrec_get_use_line
symrec.h, 159
yasm_symrec_get_visibility
symrec.h, 159
yasm_symrec_is_abs
symrec.h, 159
yasm_symrec_is_curpos
symrec.h, 159
yasm_symrec_is_special
symrec.h, 160
yasm_symrec_print
symrec.h, 160
yasm_symrec_set_common_size
symrec.h, 160
yasm_symrec_set_objext_valparams
symrec.h, 160
yasm_symrec_set_segment
symrec.h, 160
yasm_symrec_set_size
symrec.h, 161
yasm_symtab
coretype.h, 70

yasm_symtab_abs_sym
symrec.h, 161
yasm_symtab_create
symrec.h, 161
yasm_symtab_declare
symrec.h, 161
yasm_symtab_define_curpos
symrec.h, 162
yasm_symtab_define_equ
symrec.h, 162
yasm_symtab_define_label
symrec.h, 162
yasm_symtab_define_special
symrec.h, 163
yasm_symtab_destroy
symrec.h, 163
yasm_symtab_first
symrec.h, 163
yasm_symtab_get
symrec.h, 163
yasm_symtab_iter
symrec.h, 154
yasm_symtab_iter_value
symrec.h, 164
yasm_symtab_next
symrec.h, 164
yasm_symtab_parser_finalize
symrec.h, 164
yasm_symtab_print
symrec.h, 164
yasm_symtab_set_case_sensitive
symrec.h, 165
yasm_symtab_traverse
symrec.h, 165
yasm_symtab_traverse_callback
symrec.h, 154
yasm_symtab_use
symrec.h, 165
yasm_unescape_cstring
file.h, 112
yasm_valparam, 34
coretype.h, 70
id_prefix, 35
yasm_param_type, 35
yasm_valparam::yasm_param, 28
yasm_valparamhead
coretype.h, 70
yasm_value, 35
abs, 36
coretype.h, 71
curpos_rel, 36
ip_rel, 36
jump_target, 36
no_warn, 37
rel, 37
rshift, 37
section_rel, 37
seg_of, 37
sign, 37
size, 37
wrt, 37
yasm_value_delete
value.h, 175
yasm_value_finalize
value.h, 175
yasm_value_finalize_expr
value.h, 175
yasm_value_get_intnum
value.h, 176
yasm_value_init_copy
value.h, 176
yasm_value_init_sym
value.h, 176
yasm_value_initialize
value.h, 176
yasm_value_output_basic
value.h, 177
yasm_value_print
value.h, 177
yasm_value_set_curpos_rel
value.h, 177
yasm_vp_create_expr
valparam.h, 171
yasm_vp_create_id
valparam.h, 171
yasm_vp_create_string
valparam.h, 171
yasm_vp_expr
valparam.h, 172
yasm_vp_id
valparam.h, 172
yasm_vp_string
valparam.h, 172
yasm_vps_append
valparam.h, 172
yasm_vps_create
valparam.h, 173
yasm_vps_delete
valparam.h, 173
yasm_vps_destroy
valparam.h, 173
yasm_vps_first
valparam.h, 173
yasm_vps_initialize
valparam.h, 173
yasm_vps_next
valparam.h, 173
yasm_vps_print

valparam.h, 174
yasm_warn_class
errwarn.h, 79, 80
yasm_warn_clear
errwarn.h, 83
yasm_warn_disable
errwarn.h, 83
yasm_warn_disable_all
errwarn.h, 84
yasm_warn_enable
errwarn.h, 84
yasm_warn_fetch
errwarn.h, 84
yasm_warn_occurred
errwarn.h, 84
yasm_warn_set
errwarn.h, 84
yasm_warn_set_va
errwarn.h, 84
yasm_xcalloc
coretype.h, 73
yasm_xfree
coretype.h, 74
yasm_xmalloc
coretype.h, 74
yasm_xrealloc
coretype.h, 74