

My Project

Generated by Doxygen 1.9.4

1 Class Index	1
1.1 Class List	1
2 File Index	3
2.1 File List	3
3 Class Documentation	5
3.1 assembly_code Struct Reference	5
3.1.1 Detailed Description	5
3.1.2 Member Data Documentation	5
3.1.2.1 code	5
3.1.2.2 position	5
3.1.2.3 size	6
3.2 cvt_u_int64_t_int Union Reference	6
3.2.1 Detailed Description	6
3.2.2 Member Data Documentation	6
3.2.2.1 extended_address	6
3.2.2.2 rel_addr	6
3.3 label_table Struct Reference	7
3.4 opcode Struct Reference	7
3.4.1 Detailed Description	7
3.4.2 Member Data Documentation	7
3.4.2.1 code	7
3.4.2.2 size	7
3.5 stack Struct Reference	8
3.6 stack_node Struct Reference	8
4 File Documentation	9
4.1 label_table.h	9
4.2 log.h	10
4.3 src/translator.h File Reference	11
4.3.1 Detailed Description	13
4.3.2 Macro Definition Documentation	14
4.3.2.1 HOST_MEMORY_COUNT	14
4.3.2.2 MIN_DST_CODE_SIZE	14
4.3.2.3 PAGESIZE	14
4.3.2.4 TRANSLATE_PUSH_SIZE	14
4.3.2.5 UNKNOWN	14
4.3.3 Enumeration Type Documentation	14
4.3.3.1 HOST_ASSEMBLY_REG_ID	14
4.3.3.2 HOST_STACK_OP_CODES	15
4.3.3.3 REG_MASK	15
4.3.3.4 TRANSLATION_ERROR	15

4.3.3.5 VCMPPD_COMPARISONS_CODE	15
4.3.3.6 WORD_SIZE	15
4.3.3.7 X86_ASSEMBLY_OPCODES	15
4.3.3.8 X86_ASSEMBLY_OPCODES_SIZE	15
4.3.4 Function Documentation	15
4.3.4.1 assembly_code_aligned_init()	15
4.3.4.2 assembly_code_init()	16
4.3.4.3 load_code()	16
4.3.4.4 make_label_table()	17
4.4 translator.h	17

Index	23
--------------	-----------

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

assembly_code	5
cvt_u_int64_t_int	6
label_table	7
opcode	7
stack	8
stack_node	8

Chapter 2

File Index

2.1 File List

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

lib/ label_table.h	9
lib/ log.h	10
src/ translator.h The header of translator, containing all used functions in JIT	11

Chapter 3

Class Documentation

3.1 `assembly_code` Struct Reference

```
#include <translator.h>
```

Public Attributes

- `char * code`
- `int position`
- `size_t size`

3.1.1 Detailed Description

Structure containing the assembled code and host (source) assembler code

3.1.2 Member Data Documentation

3.1.2.1 `code`

```
char* assembly_code::code
```

Buffer to contation code

3.1.2.2 `position`

```
int assembly_code::position
```

position of current opcode in buffer

3.1.2.3 size

```
size_t assembly_code::size
```

size of buffer

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

- [src/translator.h](#)

3.2 cvt_u_int64_t_int Union Reference

```
#include <translator.h>
```

Public Attributes

- int [rel_addr](#)
- u_int64_t [extended_address](#)

3.2.1 Detailed Description

Union to zero-extended conversation from int to u_int64_t

3.2.2 Member Data Documentation

3.2.2.1 extended_address

```
u_int64_t cvt_u_int64_t_int::extended_address
```

Zero extended value.

3.2.2.2 rel_addr

```
int cvt_u_int64_t_int::rel_addr
```

Value to convert. The name of the variable is because it is used primarily for address translation.

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

- [src/translator.h](#)

3.3 label_table Struct Reference

Public Attributes

- int **size**
- [stack](#) * **elems**

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

- lib/label_table.h

3.4 opcode Struct Reference

```
#include <translator.h>
```

Public Attributes

- u_int64_t **code**
- int **size**

3.4.1 Detailed Description

Structure used to simplify writing opcodes

3.4.2 Member Data Documentation

3.4.2.1 code

```
u_int64_t opcode::code
```

Opcode, that should be written in the buffer

3.4.2.2 size

```
int opcode::size
```

Size of this opcode

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

- src/[translator.h](#)

3.5 `stack` Struct Reference

Public Attributes

- `int` **size**
- `int` **capacity**
- `stack_node` * **data**

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

- `lib/label_table.h`

3.6 `stack_node` Struct Reference

Public Attributes

- `int` **label**
- `int` **jmp**
- `size_t` **code_pos**

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

- `lib/label_table.h`

Chapter 4

File Documentation

4.1 label_table.h

```
1 #if !defined LABEL_TABLE_INCLUDED
2
3 #include <stdint>
4 #include <stdlib.h>
5 #include <immintrin.h>
6 #define LABEL_TABLE_INCLUDED
7
8 #define HT_ERROR -1
9 #define NOT_FOUND -1
10
11 #define CYCLE 1
12 // 128 * 16 * 32 * 16
13 // TODO: Rework for dynamic value of size
14 #define MIN_STACK_SIZE 16
15 #define MIN_LABEL_TABLE_SIZE 128
16
17 #define STACK self->elems
18
19
20 struct stack_node
21 {
22     int label; //
23     int jmp;
24     size_t code_pos;
25 };
26
27 struct stack
28 {
29     int size;
30     int capacity;
31     stack_node* data;
32 };
33
34 struct label_table // Hash table for immediate value
35 {
36     int size;
37     stack* elems;
38 };
39
40
41 //+=====| STACK |======+
42
43
44 void stack_init(stack* const __restrict stack, const size_t size)
45     __attribute__((nonnull(1)));
46
47
48 int stack_push(stack* const __restrict stack,
49               const int key,
50               const int data)
51     __attribute__((nonnull(1)));
52
53 int stack_destr(stack* const __restrict stack)
54     __attribute__((nonnull(1)));
55 //-----
56
57
58 int label_table_init(label_table* const __restrict self)
```

```

59     __attribute__((nonnull(1)));
60
61 void label_table_manual_destr(label_table* const __restrict self)
62     __attribute__((nonnull(1)));
63
64
65 void label_table_add(label_table* const __restrict self,
66                     const int key,
67                     const int data)
68     __attribute__((always_inline, nonnull(1)));
69
70
71 void set_all_cycles(label_table* self,
72                    const int indx,
73                    const size_t code_pos,
74                    const int label_pos)
75     __attribute__((nonnull(1)));
76
77 inline int label_table_search_by_label(label_table* const __restrict self,
78                                       const int label_pos)
79     __attribute__((always_inline, hot));
80
81
82 size_t get_code_pos_by_jump(label_table* const __restrict self,
83                             const int label_pos,
84                             const int jmp_pos)
85     __attribute__((hot, nonnull(1)));
86
87 size_t* get_code_pos_ptr_by_jump(label_table* const __restrict self,
88                                  const int label_pos,
89                                  const int jmp_pos)
90     __attribute__((hot, nonnull(1)));
91
92 void label_table_destr(label_table* const __restrict self)
93     __attribute__((nonnull(1)));
94
95 inline void label_table_add(label_table* const __restrict self,
96                             const int label,
97                             const int jmp)
98 {
99     int indx = _mm_crc32_u32(label, 0xDEDED) % MIN_LABEL_TABLE_SIZE;
100    // printf("indx =%d\n", indx);
101    stack_push(&self->elems[indx], label, jmp);
102 }
103
104 inline int label_table_search_by_label(label_table* const __restrict self,
105                                       const int label_pos)
106 {
107     int indx = _mm_crc32_u32(label_pos, 0xDEDED) % MIN_LABEL_TABLE_SIZE;
108
109     if (STACK[indx].size == 0) {
110         return NOT_FOUND;
111     }
112
113     return indx;
114 };
115 };
116
117
118
119
120 #endif

```

4.2 log.h

```

1 #ifndef LOG_INCLUDED
2 #define LOG_INCLUDED
3 #include <stdio.h>
4 #include <stdarg.h>
5 #include <stdlib.h>
6 #include <execinfo.h>
7
8 #define RED "\u001b[31m"
9
10 #define FATAL_RED "\u001b[31;1m"
11 #define GREEN "\u001b[32m"
12 #define YELLOW "\u001b[33m"
13 #define BLUE "\u001b[34m"
14 #define MAGENTA "\u001b[35m"
15 #define CYAN "\u001b[36m"
16 #define END "\u001b[0m"
17
18 #define ERROR(condition, ret_val,...) \

```

```

19     if(condition) {
20         ErrorPrint(__VA_ARGS__);
21         return ret_val;
22     }
23
24 #define RET_IF(condition, ret_val) \
25 if(condition) { \
26     return ret_val; \
27 }
28
29 void SetLogFile(FILE* log_file = nullptr);
30
31 void ResetLogFile();
32
33 void ResetAllLogFiles();
34
35 int PrintToLog(const char* format, ...);
36
37 FILE* GetCurrentLogFile();
38
39 #define ErrorPrint(...) \
40 ErrorPrint_(__PRETTY_FUNCTION__, __LINE__, __FILE__, __VA_ARGS__);
41
42 #define PrettyPrint(...) \
43 PrettyPrint_(__PRETTY_FUNCTION__, __LINE__, __FILE__, __VA_ARGS__);
44
45 int PrettyPrint_(const char* function, const int line, const char* file, const char* format, ...);
46
47 int ErrorPrint_(const char* function, const int line, const char* file, const char* format, ...);
48
49 #endif

```

4.3 src/translator.h File Reference

The header of translator, containing all used functions in JIT.

```

#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include <log.h>
#include <sys/types.h>
#include <sys/mman.h>
#include <label_table.h>

```

Classes

- struct [opcode](#)
- union [cvt_u_int64_t_int](#)
- struct [assembly_code](#)

Macros

- #define [PAGESIZE](#) 4096
- #define [HOST_MEMORY_COUNT](#) 993
- #define [TRANSLATE_PUSH_SIZE](#) 23
- #define [UNKNOWN](#) 0
- #define [MIN_DST_CODE_SIZE](#) 2 << 14
- #define [BYTE](#)(val) val * 8
- #define [OPSIZE](#)(op_code_name) SIZEOF_##op_code_name

Some usefull defines to decrease amount of code.

Enumerations

- enum [WORD_SIZE](#) : u_int64_t { **DWORD** = 4 , **QWORD** = 8 , **XMMWORD** = 16 }
- enum [VCMPPD_COMPARISONS_CODE](#) : u_int64_t { **EQUAL** = 0 , **LESS** = 17 , **GREATER** = 30 }
- enum [REG_MASK](#) : u_int64_t { **XMM0_EXTEND** = 0x44 , **XMM5_BASE** = 0x2C , **XMM5_EXTEND** = 0x6C }
- enum [TRANSLATION_ERROR](#) { **CVT_ERROR** = -0xDEd , **MALLOC_ERROR** , **FILE_OPENING_ERROR** , **UNKNOWN_FILE** }
- enum [HOST_STACK_OP_CODES](#) {
PUSH = 1 , **POP** = 2 , **IN** = 3 , **OUT** = 4 ,
MUL = 5 , **ADD** = 6 , **SUB** = 7 , **DIV** = 8 ,
HLT = 10 , **JB** = 11 , **CALL** = 12 , **RET** = 13 ,
JA = 14 , **JMP** = 15 , **SQRT** = 16 , **JE** = 17 ,
POPM = 2 | 0x70 , **POPR** = 2 | 0x60 , **POPRM** = 2 | 0x35 , **PUSHM** = 1 | 0x70 ,
PUSHR = 1 | 0x60 , **PUSHRM** = 1 | 0x35 }
- enum [HOST_ASSEMBLY_REG_ID](#) : u_int64_t { **AX** = 1 , **BX** , **CX** , **DX** }
- enum [X86_ASSEMBLY_OPCODES](#) : u_int64_t {
RAX = 0 , **RBX** = 3 , **RCX** = 1 , **RDx** = 2 ,
XMM0 = RAX , **XMM1** = RCX , **XMM2** = RDx , **XMM3** = RBX ,
XMM4 , **VMOVQ_XMM_RSP_IMM** = 0x0024007EFAC5 , **VMOVQ_RSP_XMM** = 0x2400D6F9C5 , **SUB_RSP_IMM** = 0x00EC8348 ,
ADD_RSP_IMM = 0x00C48348 , **VMOVQ_RSP_IMM_XMM** = 0x002400D6F9C5 , **ADDSD_XMM_RSP_IMM** = 0x002400580FF2 , **SUBSD_XMM_RSP_IMM** = 0x0024005C0FF2 ,
MULSD_XMM_RSP_IMM = 0x002400590FF2 , **DIVSD_XMM_RSP_IMM** = 0x0024005E0FF2 , **VSQRTPD_XMM0_XMM0** = 0xC051F9C5 , **VMOVQ_XMM5_R13_B_IMM** = 0x006D7E7AC1C4 ,
VMOVQ_XMM5_R13_D_IMM = 0xAD7E7AC1C4 , **VMOVQ_R13_B_IMM_XMM5** = 0x006DD679C1C4 ,
VMOVQ_R13_D_IMM_XMM5 = 0xADD679C1C4 , **JMP_REL32** = 0x00000000E9 ,
JE_REL32 = 0x00000000840F , **VCMPPD_XMM5_XMM0_XMM5** = 0x00EDC2F9C5 , **CMP_R14D_3** = 0x03FE8341 , **CMP_R14D_1** = 0x01FE8341 ,
LEA_RDI_RSP_16 = 0x10247C8D48 , **MOV_EDI_0** = 0x00000000BF , **MOVMSKPD_R14D_XMM5** = 0xF5500F4466 , **MOV_RDI_RSP** = 0xE78948 ,
NATIVE_RET = 0xC3 , **NATIVE_CALL** = 0x00000000E8 , **MOV_R15_RSP** = 0xE78949 , **MOV_RSP_R15** = 0xFC894C ,
MOV_R14 = 0xBE49 , **MOV_TO_STACK_R14** = 0x2434894C , **MOV_R13** = 0xBD49 }
- enum [X86_ASSEMBLY_OPCODES_SIZE](#) {
sizeof_VMOVQ_RSP_XMM = 5 , **sizeof_VMOVQ_RSP_IMM_XMM** = 6 , **sizeof_SUB_RSP_IMM** = 4 ,
sizeof_ADD_RSP_IMM = 4 ,
sizeof_ADDSD_XMM_RSP_IMM = 6 , **sizeof_MOV_R15_RSP** = 3 , **sizeof_MOV_RSP_R15** = 3 ,
sizeof_NATIVE_CALL = 5 ,
sizeof_MOV_R14 = 2 , **sizeof_MOV_TO_STACK_R14** = 4 , **sizeof_VMOVQ_XMM_RSP_IMM** = 6 ,
sizeof_VMOVQ_XMM5_R13_B_IMM = 6 ,
sizeof_VMOVQ_XMM5_R13_D_IMM = 5 , **sizeof_MOV_R13** = 2 , **sizeof_JMP_REL32** = 5 , **sizeof_JE_REL32** = 6 ,
sizeof_VCMPPD_XMM5_XMM0_XMM5 = 5 , **sizeof_MOVMSKPD_R14D_XMM5** = 5 , **sizeof_CMP_R14D_1** = 4 , **sizeof_CMP_R14D_3** = 4 ,
sizeof_MOV_RDI_RSP = 3 , **sizeof_MOV_EDI_0** = 5 , **sizeof_VSQRTPD_XMM0_XMM0** = 4 ,
sizeof_LEA_RDI_RSP_16 = 5 ,
sizeof_VMOVQ_R13_B_IMM_XMM5 = 6 , **sizeof_VMOVQ_R13_D_IMM_XMM5** = 5 , **sizeof_RET** = 1
}

Functions

- int [assembly_code_init](#) ([assembly_code](#) *const __restrict self, const size_t size) __attribute__((nonnull(1)))
function to initialize [assembly_code](#) structure
- int [load_code](#) (const char *const __restrict src_file_name, [assembly_code](#) *const __restrict src_code_save) __attribute__((nonnull(1)))
function to read code from file and load it to struct field of [assembly_code](#)

- int void **make_label_table** (**assembly_code** *const __restrict src_code, **label_table** *const __restrict table) __attribute__((nonnull(1)))
function for making so-called **label_table** - structure containing all info about labels (such as position in source code)
- int void **assembly_code_aligned_init** (**assembly_code** *const __restrict self, const size_t alignment, const size_t size) __attribute__((nonnull(1)))
same as **assembly_code_init()**, but the memory allocated using **aligned_alloc()** function
- void **label_setting** (**assembly_code** *const __restrict dst_code, **label_table** *const __restrict table, const int indx, const size_t code_pos, const int label_pos) __attribute__((nonnull(1)))
- void **save_jmp_n_call_rel32** (**assembly_code** *const __restrict dst_code, const size_t code_pos) __attribute__((nonnull(1)))
- **opcode** **translate_jmp_n_call** (**assembly_code** *const __restrict dst_code, const int jmp_code) __attribute__((nonnull(1)))
- void **execute_start** (char *const __restrict execution_buffer, const int time_flag) __attribute__((nonnull(1)))
- int **translation_start** (const char *const __restrict src_file_name, **assembly_code** *const __restrict dst_buffer, const int time_flag) __attribute__((nonnull(1)))
- int **load_src_assembly_code** (const char *const __restrict src_file_name, **assembly_code** *const __restrict src_code_save) __attribute__((nonnull(1)))
- int **command_line_handler** (int argc, char *argv[]) __attribute__((nonnull(2)))
- int **double_printf** (double *value)
- int **double_scanf** (double *value)
- void **write_command** (**assembly_code** *const __restrict dst_code, **opcode** operation_code) __attribute__((always_inline))
- void **nonnull** (1))
- void **translate_load_rsp** (**assembly_code** *const __restrict dst_node) __attribute__((always_inline))
- u_int64_t **cvt_host_reg_id_to_native** (const int host_reg_id, const u_int64_t suffix, const u_int64_t offset) __attribute__((always_inline))
- void **translate_push** (**assembly_code** *const __restrict dst_code, const u_int64_t data) __attribute__((nonnull(1)))
- void **translate_push_r** (**assembly_code** *const __restrict dst_code, const int reg_id) __attribute__((nonnull(1)))
- void **translate_cycle** (**assembly_code** *const __restrict dst_code, **label_table** *const __restrict table, const int label_pos, const int jmp_pos, const int jmp_code) __attribute__((nonnull(1)))
- void **translate_jmp** (**assembly_code** *const __restrict dst_code, **label_table** *const __restrict table, const int label_pos, const int jmp_pos) __attribute__((nonnull(1)))
- void **translate_ahead_jmp_n_call** (**assembly_code** *const __restrict dst_code, **label_table** *const __restrict table, const int label_pos, const int jmp_n_call_pos, const int jmp_n_call_code) __attribute__((nonnull(1)))
- void **jmp_n_call_handler** (**assembly_code** *const __restrict dst_code, **label_table** *const __restrict table, const int label_pos, const int jmp_n_call_pos, const int jmp_n_call_code) __attribute__((nonnull(1)))
- void **translate_save_rsp** (**assembly_code** *const __restrict dst_node) __attribute__((nonnull(1)))
- void **translate_stdout** (**assembly_code** *const __restrict dst_buffer) __attribute__((always_inline))
- void **translate_two_pop_for_cmp** (**assembly_code** *const __restrict dst_code, const int jmp_code) __attribute__((nonnull(1)))
- void **translate_ret** (**assembly_code** *const __restrict dst_code) __attribute__((nonnull(1)))
- void **translate_arithmetic_op** (**assembly_code** *const __restrict dst_code, const int op_id) __attribute__((nonnull(1)))

Variables

- **opcode** **always_inline**

4.3.1 Detailed Description

The header of translator, containing all used functions in JIT.

4.3.2 Macro Definition Documentation

4.3.2.1 HOST_MEMORY_COUNT

```
#define HOST_MEMORY_COUNT 993
```

All memory indexes that used in host assembler

4.3.2.2 MIN_DST_CODE_SIZE

```
#define MIN_DST_CODE_SIZE 2 << 14
```

Size of execution buffer (currently this is fixed value)

4.3.2.3 PAGESIZE

```
#define PAGESIZE 4096
```

Default linux page size. Needed to mprotect correct work

4.3.2.4 TRANSLATE_PUSH_SIZE

```
#define TRANSLATE_PUSH_SIZE 23
```

Size of opcode of one push register, needed to correct call

4.3.2.5 UNKNOWN

```
#define UNKNOWN 0
```

Define to mark unknown yet label

4.3.3 Enumeration Type Documentation

4.3.3.1 HOST_ASSEMBLY_REG_ID

```
enum HOST_ASSEMBLY_REG_ID : u_int64_t
```

My own register "opcodes" (masks to be exact)

4.3.3.2 HOST_STACK_OP_CODES

enum [HOST_STACK_OP_CODES](#)

Opcodes for my CPU emulator (soft-CPU)

4.3.3.3 REG_MASK

enum [REG_MASK](#) : `u_int64_t`

Enum containing register code in x86 Assembler

4.3.3.4 TRANSLATION_ERROR

enum [TRANSLATION_ERROR](#)

Enum for error handling

4.3.3.5 VCMPPD_COMPARISONS_CODE

enum [VCMPPD_COMPARISONS_CODE](#) : `u_int64_t`

Enum containing comparison "code" for VCMPPD instruction

4.3.3.6 WORD_SIZE

enum [WORD_SIZE](#) : `u_int64_t`

Enum containing word sizes

4.3.3.7 X86_ASSEMBLY_OPCODES

enum [X86_ASSEMBLY_OPCODES](#) : `u_int64_t`

Enum containing all x86 opcodes

4.3.3.8 X86_ASSEMBLY_OPCODES_SIZE

enum [X86_ASSEMBLY_OPCODES_SIZE](#)

Enum containing the size (in bytes) of each x86 opcodes, used by program

4.3.4 Function Documentation

4.3.4.1 assembly_code_aligned_init()

```
int void int assembly_code_aligned_init (
    assembly\_code *const __restrict self,
    const size_t alignment,
    const size_t size )
```

same as [assembly_code_init\(\)](#), but the memory allocated using `aligned_alloc()` function

Parameters

<i>self</i>	object to initialize
<i>alignment</i>	alignment of allocated memory
<i>size</i>	size of allocated memory for structure field <code>self->buffer</code>

4.3.4.2 assembly_code_init()

```
int assembly_code_init (
    assembly\_code *const __restrict self,
    const size_t size )
```

function to initialize [assembly_code](#) structure

Parameters

<i>self</i>	object to initialize
<i>size</i>	size of allocated memory for structure field <code>self->buffer</code>

Returns

return `MALLOC_ERROR` (see `TRANSLATION_ERROR`) if allocation is failed. Return 0 if succeed.

See also

[TRANSLATION_ERROR](#)

4.3.4.3 load_code()

```
int load_code (
    const char *const __restrict src_file_name,
    assembly\_code *const __restrict src_code_save )
```

function to read code from file and load it to struct field of [assembly_code](#)

Parameters

<i>src_file_name</i>	name of source file to read
<i>src_code_save</i>	object of structure assembly_code to save source code

4.3.4.4 make_label_table()

```
int void make_label_table (
    assembly_code *const __restrict src_code,
    label_table *const __restrict table )
```

function for making so-called [label_table](#) - structure contating all info about labels (such as position in source code)

Parameters

src_code	source code where labels will be looked up
table	object of label_table to save info about labels

4.4 translator.h

[Go to the documentation of this file.](#)

```
1
6 #include <stdlib.h>
7 #include <string.h>
8 #include <stdio.h>
9 #include <log.h>
10 #include <sys/types.h>
11 #include <sys/mman.h>
12 #include <label_table.h>
13
14 #define PAGESIZE 4096
15 #define HOST_MEMORY_COUNT 993
16 #define TRANSLATE_PUSH_SIZE 23
17 #define UNKNOWN 0
18 #define MIN_DST_CODE_SIZE 2 « 14
24 #define BYTE(val) val * 8
25 #define OPSIZE(op_code_name) sizeof_##op_code_name
26
27
28 //+=====| WORD SIZES |=====+
29
32 enum WORD_SIZE : u_int64_t {
33     DWORD = 4,
34     QWORD = 8,
35     XMMWORD = 16,
36 };
37
38 //-----
39
40 //+=====| COMPARISONS |=====+
41
44 enum VCMPPD_COMPARISONS_CODE : u_int64_t {
45     EQUAL = 0,
46     LESS = 17,
47     GREATER = 30
48 };
49
50 //-----
51
52
53 //+=====| MASK FOR XMM REGISTER |=====+
54
57 enum REG_MASK : u_int64_t {
58     XMM0_EXTEND = 0x44,
59     XMM5_BASE = 0x2C,
60     XMM5_EXTEND = 0x6C
61 };
62
63 //-----
64
65
66 //+=====| TRANSLATION ERRORS |=====+
67
70 enum TRANSLATION_ERROR {
71     CVT_ERROR = -0xDEd,
72     MALLOC_ERROR,
73     FILE_OPENING_ERROR,
```

```

74 UNKNOWN_FILE
75 };
76
77
78 //+=====| HOST ASSEMBLER OPCODES |=====+
79
82 enum HOST_STACK_OP_CODES { // C++11
83     PUSH = 1,
84     POP = 2,
85     IN = 3,
86     OUT = 4,
87     MUL = 5,
88     ADD = 6,
89     SUB = 7,
90     DIV = 8,
91     HLT = 10,
92     JB = 11,
93     CALL = 12,
94     RET = 13,
95     JA = 14,
96     JMP = 15,
97     SQRD = 16,
98     JE = 17,
99     POPM = 2 | 0x70,
100     POPR = 2 | 0x60,
101     POPRM = 2 | 0x35, // Pop to memory with register index
102     PUSHM = 1 | 0x70,
103     PUSHR = 1 | 0x60,
104     PUSHRM = 1 | 0x35
105
106 };
107
110 enum HOST_ASSEMBLY_REG_ID : u_int64_t {
111     AX = 1,
112     BX,
113     CX,
114     DX
115 };
116
117
118 //-----
119
120 //+=====| NATIVE ASSEMBLER |=====+
121
124 enum X86_ASSEMBLY_OPCODES : u_int64_t {
125
126     //+=====| REG_IDS |=====+
127     //
128     RAX = 0,
129     RBX = 3, //
130     RCX = 1,
131     RDX = 2,
132
133     XMM0 = RAX,
134     XMM1 = RCX,
135     XMM2 = RDX,
136     XMM3 = RBX,
137     XMM4,
138
139     //-----
140
141     //+=====| OP_CODES |=====+
142
143     // NOT THAT ALL OP_CODES REVERSED BECAUSE OF LITTLE ENDIAN
144     /*
145         +-----+
146         // .-----v v v
147         " . vmovq xmm(0-5), [rsp - 8] | XMM Encoding
148         || | +-----^ | xmm0 = 0x44 = 01000100b
149         ^^ ^ ^ | xmm1 = 0x4C = 01010100b ... */
150 VMOVQ_XMM_RSP_IMM = 0x0024007EFAC5, //
151     /*
152         +-----+
153         / +-----+ | XMM Encoding:
154         . / | | | | xmm0 = 0x4
155         | . V v | xmm1 = 0xC ..
156         | | vmovq [rsp], xmm(0-5)
157         ^ ^
158 VMOVQ_RSP_XMM = 0x2400D6F9C5, //
159     /*
160         +-----+
161         / |
162         . V
163         | sub rsp, imm
164         ^
165 SUB_RSP_IMM = 0x00EC8348, //
166 ADD_RSP_IMM = 0x00C48348, //

```

```

167      /*
168          .-----+-----+-----< XMM Encoding: see +++
169          /  -----+-----+-----
170          | /      V      V      V
171          | | vmovq [rsp + 00], xmm(0-4)
172          | | .-----^
173          V V V
174      VMOVQ_RSP_IMM_XMM = 0x002400D6F9C5, //
175      /*
176          ^ ^ ^
177          | | |
178          vmovq code
179          .-----+-----+-----< Encoding: See ++
180          /  .-----+-----+-----
181          ./ ./      V      V      V
182          | | addsd [rsp + 00], xmm(0-4)
183          | | .-----^
184          | | /
185      ADDSD_XMM_RSP_IMM = 0x002400580FF2,
186      /*
187          ^ ^ ^
188          | | |
189          addsd
190      SUBSD_XMM_RSP_IMM = 0x0024005C0FF2,
191      MULSD_XMM_RSP_IMM = 0x002400590FF2,
192      DIVSD_XMM_RSP_IMM = 0x0024005E0FF2,
193      /*
194          +-----+
195          / / / /
196          . . . .
197          | | | |
198          ^ ^ ^ ^
199          vsqrtpd xmm0, xmm0
200      VSQRTPD_XMM0_XMM0 = 0xC051F9C5,
201      /*
202          Used to memory reference
203          +-----+
204          /
205          .
206          |
207          ^
208          vmovq xmm5, [r13 + imm]
209      VMOVQ_XMM5_R13_B_IMM = 0x006D7E7AC1C4,
210      /*
211          Identical, but the imm is double word (4 bytes)
212      VMOVQ_XMM5_R13_D_IMM = 0xAD7E7AC1C4,
213      VMOVQ_R13_B_IMM_XMM5 = 0x006DD679C1C4,
214      /*
215      !NOTE!: SIZEOF VMOVQ_R13_D_IMM_XMM is 9 bytes,
216      so i have to split it, and cannot use the last byte.
217      It is suit for my case
218      */
219      VMOVQ_R13_D_IMM_XMM5 = 0xADD679C1C4,
220      /* Relative jump
221          +-----+
222          / / / /
223          . . . .
224          | | | |
225          ^ ^ ^ ^
226          jmp rel32
227      JMP_REL32 = 0x00000000E9,
228      JE_REL32 = 0x00000000840F,
229      /*
230          +-----+
231          /
232          .
233          |
234          ^
235          vcmppd xmm5, xmm0, xmm5, 0 - Comparison mod
236      VCMPPD_XMM5_XMM0_XMM5 = 0x00EDC2F9C5,
237      /*
238          +-----+
239          / / / /
240          . . . .
241          | | | |
242          | | | |
243          | | | |
244          | | | |
245          use
246          ^ ^ ^ ^
247      CMP_R14D_3 = 0x03FE8341,
248      CMP_R14D_1 = 0x01FE8341,
249      LEA_RDI_RSP_16 = 0x10247C8D48,
250      /*
251          +-----+

```

```

253          / / / /          |
254          . . .          V
255          | | | |          mov edi, 0
256          ^ ^ ^ ^
257      MOV_EDI_0 = 0x00000000BF,
258
259      /*
260          +---+---+---+---+
261          / / / /          |
262          . . . .          +---+---+---+---+---+
263          | | | |          v v v v v v v v v v
264          ^ ^ ^ ^          movmskpd r14d, xmm5
265      MOVMSKPD_R14D_XMM5 = 0xF550F4466,
266      /*
267          To call scanf
268          +---+---+---+---+
269          / / /          |
270          . . .          +---+---+---+---+
271          | | |          v v v v v v
272          ^ ^ ^          mov rdi, rsp
273      MOV_RDI_RSP = 0xE78948,
274
275      NATIVE_RET = 0xC3,
276
277      /*          .-.-.-----< Rel 32 immediate address
278          | | | |
279          v v v v
280      NATIVE_CALL = 0x00000000E8,
281
282      MOV_R15_RSP = 0xE78949, // Full opcode because instruction is used once
283      MOV_RSP_R15 = 0xFC894C,
284      MOV_R14 = 0xBE49,
285      MOV_TO_STACK_R14 = 0x2434894C,
286      MOV_R13 = 0xBD49,
287
288      //-----
289  };
290
291
292  enum X86_ASSEMBLY_OPCODES_SIZE {
293      sizeof_VMOVQ_RSP_XMM = 5,
294      sizeof_VMOVQ_RSP_IMM_XMM = 6,
295      sizeof_SUB_RSP_IMM = 4,
296      sizeof_ADD_RSP_IMM = 4,
297      sizeof_ADDSD_XMM_RSP_IMM = 6,
298      sizeof_MOV_R15_RSP = 3,
299      sizeof_MOV_RSP_R15 = 3,
300      sizeof_NATIVE_CALL = 5,
301      sizeof_MOV_R14 = 2,
302      sizeof_MOV_TO_STACK_R14 = 4,
303      sizeof_VMOVQ_XMM_RSP_IMM = 6,
304      sizeof_VMOVQ_XMM5_R13_B_IMM = 6,
305      sizeof_VMOVQ_XMM5_R13_D_IMM = 5,
306      sizeof_MOV_R13 = 2,
307      sizeof_JMP_REL32 = 5,
308      sizeof_JE_REL32 = 6,
309      sizeof_VCMPD_XMM5_XMM0_XMM5 = 5,
310      sizeof_MOVMSKPD_R14D_XMM5 = 5,
311      sizeof_CMP_R14D_1 = 4,
312      sizeof_CMP_R14D_3 = 4,
313      sizeof_MOV_RDI_RSP = 3,
314      sizeof_MOV_EDI_0 = 5,
315      sizeof_VSQRTD_XMM0_XMM0 = 4,
316      sizeof_LEA_RDI_RSP_16 = 5,
317      sizeof_VMOVQ_R13_B_IMM_XMM5 = 6,
318      sizeof_VMOVQ_R13_D_IMM_XMM5 = 5,
319      sizeof_RET = 1
320  };
321
322  };
323
324  //-----
325
326  struct opcode
327  {
328      u_int64_t code;
329      int size;
330  };
331
332  union cvt_u_int64_t_int {
333      int rel_addr;
334      u_int64_t extended_address;
335  };
336
337  struct assembly_code
338  {
339      char* code;

```



```

345     int    position;
346     size_t size;
347 };
348
349 int assembly_code_init(assembly_code* const __restrict self,
350                        const size_t      size)
351     __attribute__((nonnull(1)));
352
353 int load_code(const char*    const __restrict src_file_name,
354              assembly_code* const __restrict src_code_save)
355     __attribute__((nonnull(1,2)));
356
357 //+=====| FUNCTIONS DECLARATIONS |=====+
358
359 void make_label_table(assembly_code* const __restrict src_code,
360                      label_table*  const __restrict table)
361     __attribute__((nonnull(1,2)));
362
363 int assembly_code_aligned_init(assembly_code* const __restrict self,
364                               const size_t      alignment,
365                               const size_t      size)
366     __attribute__((nonnull(1)));
367
368 void label_setting(assembly_code* const __restrict dst_code,
369                  label_table*  const __restrict table,
370                  const int      indx,
371                  const size_t   code_pos,
372                  const int      label_pos)
373     __attribute__((nonnull(1,2)));
374
375 int save_jump_n_call_rel32(assembly_code* const __restrict dst_code,
376                           const size_t      code_pos)
377     __attribute__((nonnull(1)));
378
379 opcode translate_jump_n_call(assembly_code* const __restrict dst_code,
380                             const int      jmp_code)
381     __attribute__((nonnull(1), always_inline));
382
383 void execute_start(char* const __restrict execution_buffer,
384                   const int  time_flag)
385     __attribute__((nonnull(1)));
386
387 int translation_start(const char *const __restrict src_file_name,
388                     assembly_code *const __restrict dst_buffer,
389                     const int      time_flag)
390     __attribute__((nonnull(1,2)));
391
392 int load_src_assembly_code(const char *const __restrict src_file_name,
393                          assembly_code *const __restrict src_code_save)
394     __attribute__((nonnull(1,2)));
395
396 void command_line_handler(int argc, char* argv[])
397     __attribute__((nonnull(2)));
398
399 extern "C" int double_printf(double* value);
400 extern "C" int double_scanf (double* value);
401
402 //+=====| INLINE FUNCTIONS DECLARATIONS |=====+
403
404 inline void write_command(assembly_code* const __restrict dst_code,
405                          opcode          operation_code)
406     __attribute__((always_inline, nonnull(1)));
407
408 inline void translate_load_rsp(assembly_code* const __restrict dst_node)
409     __attribute__((always_inline, nonnull(1)));
410
411 inline u_int64_t cvt_host_reg_id_to_native(const int host_reg_id,
412                                           const u_int64_t suffix,
413                                           const u_int64_t offset)
414     __attribute__((always_inline));
415
416 inline void translate_push(assembly_code* const __restrict dst_code,
417                          const u_int64_t data)
418     __attribute__((nonnull(1), always_inline));
419
420 inline void translate_push_r(assembly_code* const __restrict dst_code,
421                             const int      reg_id)
422     __attribute__((nonnull(1), always_inline));

```

```

453
454
455 inline void translate_cycle(assembly_code* const __restrict dst_code,
456                             label_table* const __restrict table,
457                             const int label_pos,
458                             const int jmp_pos,
459                             const int jmp_code)
460     __attribute__((nonnull(1,2), always_inline));
461
462 inline void translate_jmp(assembly_code* const __restrict dst_code,
463                           label_table* const __restrict table,
464                           const int label_pos,
465                           const int jmp_pos)
466     __attribute__((nonnull(1,2), always_inline));
467
468 inline void translate_ahead_jmp_n_call(assembly_code* const __restrict dst_code,
469                                        label_table* const __restrict table,
470                                        const int label_pos,
471                                        const int jmp_n_call_pos,
472                                        const int jmp_n_call_code)
473     __attribute__((nonnull(1,2), always_inline));
474
475
476 inline void jmp_n_call_handler(assembly_code* const __restrict dst_code,
477                                label_table* const __restrict table,
478                                const int label_pos,
479                                const int jmp_n_call_pos,
480                                const int jmp_n_call_code)
481     __attribute__((nonnull(1,2), always_inline));
482
483 inline void translate_save_rsp(assembly_code* const __restrict dst_code)
484     __attribute__((nonnull(1), always_inline));
485
486
487 inline void translate_stdout(assembly_code* const __restrict dst_buffer)
488     __attribute__((always_inline, nonnull(1)));
489
490
491
492 inline void translate_two_pop_for_cmp(assembly_code* const __restrict dst_code,
493                                       const int jmp_code)
494     __attribute__((nonnull(1), always_inline));
495
496
497 inline void translate_ret(assembly_code* const __restrict dst_code)
498     __attribute__((nonnull(1), always_inline));
499
500
501
502 inline void translate_arithmetic_op(assembly_code* const __restrict dst_code,
503                                     const int op_id)
504     __attribute__((nonnull(1), always_inline));
505
506 //-----
507
508

```

Index

- assembly_code, [5](#)
 - code, [5](#)
 - position, [5](#)
 - size, [5](#)
- assembly_code_aligned_init
 - translator.h, [15](#)
- assembly_code_init
 - translator.h, [16](#)
- code
 - assembly_code, [5](#)
 - opcode, [7](#)
- cvt_u_int64_t_int, [6](#)
 - extended_address, [6](#)
 - rel_addr, [6](#)
- extended_address
 - cvt_u_int64_t_int, [6](#)
- HOST_ASSEMBLY_REG_ID
 - translator.h, [14](#)
- HOST_MEMORY_COUNT
 - translator.h, [14](#)
- HOST_STACK_OP_CODES
 - translator.h, [14](#)
- label_table, [7](#)
- lib/label_table.h, [9](#)
- lib/log.h, [10](#)
- load_code
 - translator.h, [16](#)
- make_label_table
 - translator.h, [16](#)
- MIN_DST_CODE_SIZE
 - translator.h, [14](#)
- opcode, [7](#)
 - code, [7](#)
 - size, [7](#)
- PAGESIZE
 - translator.h, [14](#)
- position
 - assembly_code, [5](#)
- REG_MASK
 - translator.h, [15](#)
- rel_addr
 - cvt_u_int64_t_int, [6](#)
- size
 - assembly_code, [5](#)
 - opcode, [7](#)
- src/translator.h, [11](#), [17](#)
- stack, [8](#)
- stack_node, [8](#)
- TRANSLATE_PUSH_SIZE
 - translator.h, [14](#)
- TRANSLATION_ERROR
 - translator.h, [15](#)
- translator.h
 - assembly_code_aligned_init, [15](#)
 - assembly_code_init, [16](#)
 - HOST_ASSEMBLY_REG_ID, [14](#)
 - HOST_MEMORY_COUNT, [14](#)
 - HOST_STACK_OP_CODES, [14](#)
 - load_code, [16](#)
 - make_label_table, [16](#)
 - MIN_DST_CODE_SIZE, [14](#)
 - PAGESIZE, [14](#)
 - REG_MASK, [15](#)
 - TRANSLATE_PUSH_SIZE, [14](#)
 - TRANSLATION_ERROR, [15](#)
 - UNKNOWN, [14](#)
 - VCMPPD_COMPARISONS_CODE, [15](#)
 - WORD_SIZE, [15](#)
 - X86_ASSEMBLY_OPCODES, [15](#)
 - X86_ASSEMBLY_OPCODES_SIZE, [15](#)
- UNKNOWN
 - translator.h, [14](#)
- VCMPPD_COMPARISONS_CODE
 - translator.h, [15](#)
- WORD_SIZE
 - translator.h, [15](#)
- X86_ASSEMBLY_OPCODES
 - translator.h, [15](#)
- X86_ASSEMBLY_OPCODES_SIZE
 - translator.h, [15](#)