Contents

1 Module `Util` : This module offers some utility functions

```
module IntKey :
  sig
     type t = int
     val compare : t -> t -> int
  end
module IS :
   Set.S  with type elt = IntKey.t
module IM :
   Map.S  with type key = IntKey.t
```

`val read_lines : Pervasives.in_channel -> string list`

read each line from `in_channel`

`val range : int -> int -> int list -> int list`

tail-recursive version of python range

`val repeat : 'a -> int -> 'a list -> 'a list`

tail-recursive version of python itertools.repeat

`val get_last : 'a list -> 'a`

get the last element in a `list`

`val rm_last : 'a list -> 'a list`

remove the last element in a `list`

`val trim_1st : string -> string`

trim the 1st `char`

`val trim_last : string -> string`

trim the last `char`

`val explode : string -> char list`

split a `string` into a `list` of `chars`

`val str_rev : string -> string`

reverse the given `string`

`val split_string : string -> char -> string list`

split a `string` into a `list` of `strings`, with separator `split`

`val begins_with : string -> string -> bool` `true` if the given `string` begins with the given prefix

`val ends_with : string -> string -> bool`
`true` if the given `string` ends with the given suffix

`val contains : string -> string -> bool`
`true` if the given `string` contains the other `string`

`val common_prefix : string -> string -> string`
find the common prefix of the given `string`s

2 Module `Log` : inspired by `android.util.Log`

`val set_level : string -> unit`
set logging level

`val of_i : int -> string`
from `int` to `string`

`val v : string -> unit`
`verbose`

`val d : string -> unit`
`debug`

`val i : string -> unit`
`info`

`val w : string -> unit`
`warn`

3 Module `Java` : This module provides utility functions for handling `Java` language

3.1 Primitives

`val v : string`
`void`

`val z : string`

`boolean`

`val b : string`
`byte`

`val s : string`
`short`

`val c : string`
`char`

`val i : string`
`int`

`val j : string`
`long`

`val f : string`
`float`

`val d : string`
`double`

`val shorties : string list`
list of all short type descriptors

`val init : string`
`<init>`

`val clinit : string`
`<clinit>`

`val hashCode : string`
`hashCode`

3.2 Type Descriptions

`val is_type_descr : string -> bool`
`true` if given `string` is fit for type description

`val is_shorty_descr : string -> bool`
`true` if given `string` is fit for shorty description

`val to_type_descr : string -> string`

to type description

`val to_shorty_descr : string list -> string`
: to shorty description

`val to_java_ty : string -> string`
: from `java.lang.Object` to `Ljava/lang/Object;`

`val of_type_descr : string -> string`
: from type description

`val of_java_ty : string -> string`
: from `Ljava/lang/Object;` to `java.lang.Object`

`val is_primitive : string -> bool`
: `true` if the given type name is one of primitive types

`val is_wide : string -> bool`
: `true` if the given type name is either `long` or `double`

`val get_package_name : string -> string`
: from `Ljava/lang/Object;` to `java.lang`

`val get_class_name : string -> string`
: from `Ljava/lang/Object;` to `Object`

`val is_inner_class : string -> bool`
: `true` if the given type name is an inner class, such as ...$1

`val get_owning_class : string -> string`
: from `Lpkg/cls$n;` to `Lpkg/cls;`

3.3 Libraries

```
module Lang :
  sig
```

```
    val obj : string
```

`java.lang.Object`

```
    val cls : string
```

`java.lang.Class`

```
    val pkg : string
    java.lang.Package

val sys : string

    java.lang.System

val str : string

    java.lang.String

val sbd : string

    java.lang.StringBuilder

val thd : string

    java.lang.Thread

val thr : string

    java.lang.Throwable

val stk : string

    java.lang.StackTraceElement

val c_void : string

    java.lang.Void

val c_bool : string

    java.lang.Boolean

val c_byte : string

    java.lang.Byte

val c_short : string

    java.lang.Short

val c_char : string

    java.lang.Character

val c_int : string

    java.lang.Integer

val c_long : string

    java.lang.Long
```

```
val c_float : string
    java.lang.Float
val c_doubl : string
    java.lang.Double
val get_cls : string
    getClass
val get_name : string
    getName
val get_stk : string
    getStackTrace
val to_s : string
    toString
val append : string
    append
val start : string
    start
val run : string
    run
val concat : string
    concat
val lower : string
    toLowerCase
val upper : string
    toUpperCase
val _format : string
    format
val v_of : string
```

```
          valueOf

      val wrappers : unit -> string list
```
all wrapper classes for primitive types

```
  end

module IO :
  sig

      val ps : string

          java.io.PrintStream

  end

module Net :
  sig

      val isoc : string

          java.net.InetSocketAddress

  end

module Apache :
  sig

      val uri_reqs : unit -> string list
```
classs that implement `org.apache.http.client.methods.HttpUriRequest`

```
  end

val is_library : string -> bool
```
`true` if given class is `Java` library

4 Module `Instr` : This module defines types for Dalvik bytecodes and provides utility functions for generating, parsing and dumping instructions.

4.1 Types

```
type offset = int32
```
An address space of DEX is 32-bits

```
module IM :
   Map.S  with type key = Int32.t
```

The data pool is a mapping from 32-bit offset to `Dex.data_item`[5.1].

All other modules after this module will use this declaration.

```
type instr = opcode * operand list
```

An instruction is composed of `Instr.opcode`[4.1] and a `list` of `Instr.operand`[4.1]s.

```
type operand =
  | OPR_CONST of int64
```

constant

```
  | OPR_REGISTER of int
```

register

```
  | OPR_INDEX of int
```

index

```
  | OPR_OFFSET of offset
```

offset

Operand for `Instr.instr`[4.1]

```
type opcode =
  | OP_NOP
```

0x00

```
  | OP_MOVE
```

0x01

```
  | OP_MOVE_FROM16
```

0x02

```
  | OP_MOVE_16
```

0x03

```
  | OP_MOVE_WIDE
```

0x04

```
  | OP_MOVE_WIDE_FROM16
```

0x05

```
  | OP_MOVE_WIDE_16
```

0x06

```
  | OP_MOVE_OBJECT
```

0x07

```
  | OP_MOVE_OBJECT_FROM16
```

0x08

```
  | OP_MOVE_OBJECT_16
```

0x09

| `OP_MOVE_RESULT`
  0x0a

| `OP_MOVE_RESULT_WIDE`
  0x0b

| `OP_MOVE_RESULT_OBJECT`
  0x0c

| `OP_MOVE_EXCEPTION`
  0x0d

| `OP_RETURN_VOID`
  0x0e

| `OP_RETURN`
  0x0f

| `OP_RETURN_WIDE`
  0x10

| `OP_RETURN_OBJECT`
  0x11

| `OP_CONST_4`
  0x12

| `OP_CONST_16`
  0x13

| `OP_CONST`
  0x14

| `OP_CONST_HIGH16`
  0x15

| `OP_CONST_WIDE_16`
  0x16

| `OP_CONST_WIDE_32`
  0x17

| `OP_CONST_WIDE`
  0x18

| `OP_CONST_WIDE_HIGH16`
  0x19

| `OP_CONST_STRING`
  0x1a

| `OP_CONST_STRING_JUMBO`
  0x1b

| `OP_CONST_CLASS`
  0x1c

| `OP_MONITOR_ENTER`
  0x1d

| `OP_MONITOR_EXIT`
  0x1e

| `OP_CHECK_CAST`
  0x1f

| `OP_INSTANCE_OF`
  0x20

| `OP_ARRAY_LENGTH`
  0x21

| `OP_NEW_INSTANCE`
  0x22

| `OP_NEW_ARRAY`
  0x23

| `OP_FILLED_NEW_ARRAY`
  0x24

| `OP_FILLED_NEW_ARRAY_RANGE`
  0x25

| `OP_FILL_ARRAY_DATA`
  0x26

| `OP_THROW`
  0x27

| `OP_GOTO`
  0x28

| `OP_GOTO_16`
  0x29

| `OP_GOTO_32`
  0x2a

| `OP_PACKED_SWITCH`
  0x2b

| `OP_SPARSE_SWITCH`

0x2c

| `OP_CMPL_FLOAT`
  0x2d

| `OP_CMPG_FLOAT`
  0x2e

| `OP_CMPL_DOUBLE`
  0x2f

| `OP_CMPG_DOUBLE`
  0x30

| `OP_CMP_LONG`
  0x31

| `OP_IF_EQ`
  0x32

| `OP_IF_NE`
  0x33

| `OP_IF_LT`
  0x34

| `OP_IF_GE`
  0x35

| `OP_IF_GT`
  0x36

| `OP_IF_LE`
  0x37

| `OP_IF_EQZ`
  0x38

| `OP_IF_NEZ`
  0x39

| `OP_IF_LTZ`
  0x3a

| `OP_IF_GEZ`
  0x3b

| `OP_IF_GTZ`
  0x3c

| `OP_IF_LEZ`
  0x3d

`| OP_AGET`

0x44

`| OP_AGET_WIDE`

0x45

`| OP_AGET_OBJECT`

0x46

`| OP_AGET_BOOLEAN`

0x47

`| OP_AGET_BYTE`

0x48

`| OP_AGET_CHAR`

0x49

`| OP_AGET_SHORT`

0x4a

`| OP_APUT`

0x4b

`| OP_APUT_WIDE`

0x4c

`| OP_APUT_OBJECT`

0x4d

`| OP_APUT_BOOLEAN`

0x4e

`| OP_APUT_BYTE`

0x4f

`| OP_APUT_CHAR`

0x50

`| OP_APUT_SHORT`

0x51

`| OP_IGET`

0x52

`| OP_IGET_WIDE`

0x53

`| OP_IGET_OBJECT`

0x54

`| OP_IGET_BOOLEAN`

0x55

| `OP_IGET_BYTE`

0x56

| `OP_IGET_CHAR`

0x57

| `OP_IGET_SHORT`

0x58

| `OP_IPUT`

0x59

| `OP_IPUT_WIDE`

0x5a

| `OP_IPUT_OBJECT`

0x5b

| `OP_IPUT_BOOLEAN`

0x5c

| `OP_IPUT_BYTE`

0x5d

| `OP_IPUT_CHAR`

0x5e

| `OP_IPUT_SHORT`

0x5f

| `OP_SGET`

0x60

| `OP_SGET_WIDE`

0x61

| `OP_SGET_OBJECT`

0x62

| `OP_SGET_BOOLEAN`

0x63

| `OP_SGET_BYTE`

0x64

| `OP_SGET_CHAR`

0x65

| `OP_SGET_SHORT`

0x66

| `OP_SPUT`
  0x67

| `OP_SPUT_WIDE`
  0x68

| `OP_SPUT_OBJECT`
  0x69

| `OP_SPUT_BOOLEAN`
  0x6a

| `OP_SPUT_BYTE`
  0x6b

| `OP_SPUT_CHAR`
  0x6c

| `OP_SPUT_SHORT`
  0x6d

| `OP_INVOKE_VIRTUAL`
  0x6e

| `OP_INVOKE_SUPER`
  0x6f

| `OP_INVOKE_DIRECT`
  0x70

| `OP_INVOKE_STATIC`
  0x71

| `OP_INVOKE_INTERFACE`
  0x72

| `OP_INVOKE_VIRTUAL_RANGE`
  0x74

| `OP_INVOKE_SUPER_RANGE`
  0x75

| `OP_INVOKE_DIRECT_RANGE`
  0x76

| `OP_INVOKE_STATIC_RANGE`
  0x77

| `OP_INVOKE_INTERFACE_RANGE`
  0x78

| `OP_NEG_INT`

0x7b

| `OP_NOT_INT`
  0x7c

| `OP_NEG_LONG`
  0x7d

| `OP_NOT_LONG`
  0x7e

| `OP_NEG_FLOAT`
  0x7f

| `OP_NEG_DOUBLE`
  0x80

| `OP_INT_TO_LONG`
  0x81

| `OP_INT_TO_FLOAT`
  0x82

| `OP_INT_TO_DOUBLE`
  0x83

| `OP_LONG_TO_INT`
  0x84

| `OP_LONG_TO_FLOAT`
  0x85

| `OP_LONG_TO_DOUBLE`
  0x86

| `OP_FLOAT_TO_INT`
  0x87

| `OP_FLOAT_TO_LONG`
  0x88

| `OP_FLOAT_TO_DOUBLE`
  0x89

| `OP_DOUBLE_TO_INT`
  0x8a

| `OP_DOUBLE_TO_LONG`
  0x8b

| `OP_DOUBLE_TO_FLOAT`
  0x8c

`OP_INT_TO_BYTE`	0x8d
`OP_INT_TO_CHAR`	0x8e
`OP_INT_TO_SHORT`	0x8f
`OP_ADD_INT`	0x90
`OP_SUB_INT`	0x91
`OP_MUL_INT`	0x92
`OP_DIV_INT`	0x93
`OP_REM_INT`	0x94
`OP_AND_INT`	0x95
`OP_OR_INT`	0x96
`OP_XOR_INT`	0x97
`OP_SHL_INT`	0x98
`OP_SHR_INT`	0x99
`OP_USHR_INT`	0x9a
`OP_ADD_LONG`	0x9b
`OP_SUB_LONG`	0x9c
`OP_MUL_LONG`	0x9d
`OP_DIV_LONG`	

0x9e

| `OP_REM_LONG`
0x9f

| `OP_AND_LONG`
0xa0

| `OP_OR_LONG`
0xa1

| `OP_XOR_LONG`
0xa2

| `OP_SHL_LONG`
0xa3

| `OP_SHR_LONG`
0xa4

| `OP_USHR_LONG`
0xa5

| `OP_ADD_FLOAT`
0xa6

| `OP_SUB_FLOAT`
0xa7

| `OP_MUL_FLOAT`
0xa8

| `OP_DIV_FLOAT`
0xa9

| `OP_REM_FLOAT`
0xaa

| `OP_ADD_DOUBLE`
0xab

| `OP_SUB_DOUBLE`
0xac

| `OP_MUL_DOUBLE`
0xad

| `OP_DIV_DOUBLE`
0xae

| `OP_REM_DOUBLE`
0xaf

| `OP_ADD_INT_2ADDR`
    0xb0

| `OP_SUB_INT_2ADDR`
    0xb1

| `OP_MUL_INT_2ADDR`
    0xb2

| `OP_DIV_INT_2ADDR`
    0xb3

| `OP_REM_INT_2ADDR`
    0xb4

| `OP_AND_INT_2ADDR`
    0xb5

| `OP_OR_INT_2ADDR`
    0xb6

| `OP_XOR_INT_2ADDR`
    0xb7

| `OP_SHL_INT_2ADDR`
    0xb8

| `OP_SHR_INT_2ADDR`
    0xb9

| `OP_USHR_INT_2ADDR`
    0xba

| `OP_ADD_LONG_2ADDR`
    0xbb

| `OP_SUB_LONG_2ADDR`
    0xbc

| `OP_MUL_LONG_2ADDR`
    0xbd

| `OP_DIV_LONG_2ADDR`
    0xbe

| `OP_REM_LONG_2ADDR`
    0xbf

| `OP_AND_LONG_2ADDR`
    0xc0

| `OP_OR_LONG_2ADDR`

0xc1

| `OP_XOR_LONG_2ADDR`

0xc2

| `OP_SHL_LONG_2ADDR`

0xc3

| `OP_SHR_LONG_2ADDR`

0xc4

| `OP_USHR_LONG_2ADDR`

0xc5

| `OP_ADD_FLOAT_2ADDR`

0xc6

| `OP_SUB_FLOAT_2ADDR`

0xc7

| `OP_MUL_FLOAT_2ADDR`

0xc8

| `OP_DIV_FLOAT_2ADDR`

0xc9

| `OP_REM_FLOAT_2ADDR`

0xca

| `OP_ADD_DOUBLE_2ADDR`

0xcb

| `OP_SUB_DOUBLE_2ADDR`

0xcc

| `OP_MUL_DOUBLE_2ADDR`

0xcd

| `OP_DIV_DOUBLE_2ADDR`

0xce

| `OP_REM_DOUBLE_2ADDR`

0xcf

| `OP_ADD_INT_LIT16`

0xd0

| `OP_RSUB_INT`

0xd1

| `OP_MUL_INT_LIT16`

0xd2

| `OP_DIV_INT_LIT16`

  0xd3

| `OP_REM_INT_LIT16`

  0xd4

| `OP_AND_INT_LIT16`

  0xd5

| `OP_OR_INT_LIT16`

  0xd6

| `OP_XOR_INT_LIT16`

  0xd7

| `OP_ADD_INT_LIT8`

  0xd8

| `OP_RSUB_INT_LIT8`

  0xd9

| `OP_MUL_INT_LIT8`

  0xda

| `OP_DIV_INT_LIT8`

  0xdb

| `OP_REM_INT_LIT8`

  0xdc

| `OP_AND_INT_LIT8`

  0xdd

| `OP_OR_INT_LIT8`

  0xde

| `OP_XOR_INT_LIT8`

  0xdf

| `OP_SHL_INT_LIT8`

  0xe0

| `OP_SHR_INT_LIT8`

  0xe1

| `OP_USHR_INT_LIT8`

  0xe2

Dalvik Instruction Set, used at `Instr.instr`[4.1]

4.2 Utilities

`val of_reg : operand -> int`

unwrapping `OPR_REGISTER`

`val instr_to_string : instr -> string`

`Instr.instr`[4.1] to `string`

`val opr_to_string : operand -> string`

`Instr.operand`[4.1] to `string`

`val op_to_string : opcode -> string`

`Instr.opcode`[4.1] to `string`

`val hx_to_op_and_size : int -> opcode * int`

hex to `Instr.opcode`[4.1] and size

`val hx_to_op : int -> opcode`

hex to `Instr.opcode`[4.1]

`val op_to_hx_and_size : opcode -> int * int`

`Instr.opcode`[4.1] to hex and size

`val op_to_hx : opcode -> int`

`Instr.opcode`[4.1] to hex

`val low_reg : opcode -> int`

number of "low" registers, registers numbers higher than this must be moved to a low register before they can be used for some instructions.

```
type link_sort =
  | STRING_IDS
  | TYPE_IDS
  | FIELD_IDS
  | METHOD_IDS
  | OFFSET
  | NOT_LINK
```

sort of links in the `dex`

`val access_link : opcode -> link_sort`

which `Instr.link_sort`[4.2] does this `Instr.opcode`[4.1] access to?

`val get_argv : instr -> operand list`

retrieve actual parameters, e.g. for `invoke-*/range v0 v2 @...`, return a list of v0, v1, and v2

```
type reg_sort =
  | R_OBJ
  | R_WIDE
  | R_WIDE_L
  | R_NORMAL
```

sort of values in registers

`val get_reg_sorts : instr -> (int * reg_sort) list`

for the given `Instr.instr`[4.1], make mappings from register to its `Instr.reg_sort`[4.2]

4.3 Parsing and Dumping

`val make_instr : opcode -> int list -> instr`

build `Instr.instr`[4.1] using `Instr.opcode`[4.1] and a `list` of arguments

`val instr_to_bytes : int -> instr -> char list`

according to given base address, translate `Instr.instr`[4.1] to bytes

4.4 Generating

`val new_const : int -> int -> instr`

for given register number and constant, generate `OP_CONST`-kind `Instr.instr`[4.1]

`val new_const_id : int -> int -> int -> instr`

for a given register number, along with string or class id, generate `OP_CONST`-kind `Instr.instr`[4.1]

`val new_move : int -> int -> int -> instr`

for given destination and source registers, generate a new `OP_MOVE`-kind `Instr.instr`[4.1]

`val new_ist_of : int -> int -> int -> instr`

for given destination and source registers, along with type, generate a new `OP_INSTANCE_OF` `Instr.instr`[4.1]

`val new_obj : int -> int -> instr`

for a given destination register and type, generate a new `OP_NEW_INSTANCE` `Instr.instr`[4.1]

`val new_arr : int -> int -> int -> instr`

for a given destination register, size, and type, generate a new `OP_NEW_ARRAY` `Instr.instr`[4.1]

`val new_goto : int -> offset -> instr`

for a given `Instr.offset`[4.1], generate a new `OP_GOTO`-kind `Instr.instr`[4.1] `val new_if : int -> int -> int -> offset -> instr`

for the given test registers and `Instr.offset`[4.1], generate a new `OP_IF`-kind `Instr.instr`[4.1]

`val new_arr_op : int -> int list -> instr`

for a given value, array, index registers, generate a new `OP_A(GET|PUT)`-kind `Instr.instr`[4.1]

`val new_bin_op : int -> int list -> instr`

for the given binary op and registers, generate a new binary operation `Instr.instr`[4.1]

`val new_bin_lit_op : int -> int list -> int64 -> instr`

for the given binary op, registers, and constant, generate a new binary-lit(16|8) operation `Instr.instr`[4.1]

`val new_un_op : int -> int list -> instr`

for the given unary op and registers, generate a new unary operation `Instr.instr`[4.1]

`val new_ist_fld : int -> int -> int -> int -> instr`

for given registers and instance field id, generate `OP_I(GET|PUT)`-kind `Instr.instr`[4.1]

`val new_stt_fld : int -> int -> int -> instr`

for given register number and static field id, generate `OP_S(GET|PUT)`-kind `Instr.instr`[4.1]

`val new_invoke : int -> int list -> instr`

for given hex code and a `list` of arguments, generate `OP_INVOKE`-kind `Instr.instr`[4.1]

`val new_move_result : int -> int -> instr`

for given hex code and register number, generate `OP_MOVE_RESULT`-kind `Instr.instr`[4.1]

`val new_return : int -> int option -> instr`

for given hex code and an `option` of register, generate `OP_RETURN`-kind `Instr.instr`[4.1]

`val rv : instr`

void return

5 Module `Dex` : This module defines types for DEX binary and provides utility functions for traversing DEX file and getting info from DEX file.

5.1 Types

`exception Wrong_dex of string`

raise if something is logically incorrect

```
exception Wrong_match of string
```

raise if there is no other cases for match block

```
exception NOT_YET of string
```

raise if something is not implemented yet

```
type dex = {
  header : dex_header ;
  d_string_ids : link DynArray.t ;
  d_type_ids : link DynArray.t ;
  d_proto_ids : proto_id_item DynArray.t ;
  d_field_ids : field_id_item DynArray.t ;
  d_method_ids : method_id_item DynArray.t ;
  d_class_defs : class_def_item DynArray.t ;
  mutable d_data : data_item Instr.IM.t ;
}
```

The top-level representation of a DEX binary file

```
type link =
  | Idx of int
  | Off of Instr.offset
```

encapsulation of in/direct access

```
type dex_header = {
  magic : string ;
  checksum : int64 ;
  signature : char list ;
  mutable file_size : int ;
  header_size : int ;
  endian_tag : endian ;
  link : section ;
  map_off : link ;
  h_string_ids : section ;
  h_type_ids : section ;
  h_proto_ids : section ;
  h_field_ids : section ;
  h_method_ids : section ;
  h_class_defs : section ;
  h_data : section ;
}
```

header_item format

```
type endian =
  | LITTLE
```

ENDIAN_CONSTANT = 0x12345678

```
  | BIG
```

REVERSE_ENDIAN_CONSTANT = 0x78563412

`endian_tag` within `Dex.dex_header`[5.1]

```
type section = {
  size : int ;
  offset : link ;
}
```

a pair of size and offset, used at `Dex.dex_header`[5.1]

```
type proto_id_item = {
  shorty : link ;
  mutable return_type : link ;
  parameter_off : link ;
}
```

`Dex.proto_id_item`[5.1] appears in the `d_proto_ids`

```
type field_id_item = {
  f_class_id : link ;
  mutable f_type_id : link ;
  f_name_id : link ;
}
```

`Dex.field_id_item`[5.1] appears in the `d_field_ids`

```
type method_id_item = {
  m_class_id : link ;
  m_proto_id : link ;
  m_name_id : link ;
}
```

`Dex.method_id_item`[5.1] appears in the `d_method_ids`

```
type class_def_item = {
  c_class_id : link ;
  mutable c_access_flag : int ;
  mutable superclass : link ;
  mutable interfaces : link ;
  source_file : link ;
  annotations : link ;
  mutable class_data : link ;
  static_values : link ;
}
```

`Dex.class_def_item`[5.1] appears in the `d_class_defs`

```
type data_item =
  | MAP_LIST of map_item list
  | TYPE_LIST of link list
  | ANNO_SET_REF of link list
```
annotation_set_ref_list
```
  | ANNO_SET of link list
```
annotation_set_item
```
  | CLASS_DATA of class_data_item
  | CODE_ITEM of code_item
  | STRING_DATA of UTF8.t
```
same as `string`
```
  | DEBUG_INFO of debug_info_item
  | ANNOTATION of annotation_item
  | STATIC_VALUE of encoded_value list
```
encoded_array
```
  | ANNO_DIR of anno_dir_item
  | INSTRUCTION of Instr.instr
  | FILL_ARRAY of fill_array_data
  | SWITCH of switch
```
items in the data pool, which appears in the `d_data`

```
type map_item = {
  type_of_item : type_code ;
  mi_size : int ;
  mi_offset : link ;
}
```
`map_item` format for `map_list`, which appears in the `d_data`

```
type type_code =
  | TYPE_HEADER_ITEM
```
0x0000
```
  | TYPE_STRING_ITEM
```
0x0001
```
  | TYPE_TYPE_ID_ITEM
```
0x0002
```
  | TYPE_PROTO_ID_ITEM
```
0x0003
```
  | TYPE_FIELD_ID_ITEM
```
0x0004
```
  | TYPE_METHOD_ID_ITEM
```

0x0005

| `TYPE_CLASS_DEF_ITEM`
0x0006

| `TYPE_MAP_LIST`
0x1000

| `TYPE_TYPE_LIST`
0x1001

| `TYPE_ANNOTATION_SET_REF_LIST`
0x1002

| `TYPE_ANNOTATION_SET_ITEM`
0x1003

| `TYPE_CLASS_DATA_ITEM`
0x2000

| `TYPE_CODE_ITEM`
0x2001

| `TYPE_STRING_DATA_ITEM`
0x2002

| `TYPE_DEBUG_INFO_ITEM`
0x2003

| `TYPE_ANNOTATION_ITEM`
0x2004

| `TYPE_ENCODED_ARRAY_ITEM`
0x2005

| `TYPE_ANNOTATION_DIRECTORY_ITEM`
0x2006

type of the items, used at `Dex.map_item`[5.1]

```
type class_data_item = {
  mutable static_fields : encoded_field list ;
  mutable instance_fields : encoded_field list ;
  mutable direct_methods : encoded_method list ;
  mutable virtual_methods : encoded_method list ;
}
```

`Dex.class_data_item`[5.1] referenced from `Dex.class_def_item`[5.1]

```
type encoded_field = {
  field_idx : link ;
  f_access_flag : int ;
}
```

Dex.encoded_field[5.1] format used at Dex.class_data_item[5.1]

```
type encoded_method = {
  method_idx : link ;
  mutable m_access_flag : int ;
  code_off : link ;
}
```

Dex.encoded_method[5.1] format used at Dex.class_data_item[5.1]

```
type code_item = {
  mutable registers_size : int ;
  mutable ins_size : int ;
  mutable outs_size : int ;
  mutable tries_size : int ;
  mutable debug_info_off : link ;
  mutable insns_size : int ;
  insns : link DynArray.t ;
  mutable tries : try_item list ;
  mutable c_handlers : encoded_catch_handler list ;
}
```

Dex.code_item[5.1] referenced from Dex.encoded_method[5.1]

```
type switch = {
  mutable sw_base : link ;
  sw_size : int ;
  sw_keys : int list ;
  sw_targets : link list ;
}
```

packed-switch and sparse-switch format in insns of Dex.code_item[5.1]

```
type fill_array_data = {
  ad_width : int ;
  ad_size : int ;
  ad_data : Instr.operand list ;
}
```

fill-array-data format in insns of Dex.code_item[5.1]

```
type try_item = {
  start_addr : link ;
  end_addr : link ;
  handler_off : link ;
}
```

Dex.try_item[5.1] format referenced from Dex.code_item[5.1]

```
type encoded_catch_handler = {
  e_handlers : type_addr_pair list ;
  catch_all_addr : link ;
}
```

Dex.encoded_catch_handler[5.1] format referenced from Dex.code_item[5.1]

```
type type_addr_pair = {
  mutable ch_type_idx : link ;
  addr : link ;
}
```

encoded_type_addr_pair format referenced from Dex.encoded_catch_handler[5.1]

```
type debug_info_item = {
  line_start : int ;
  parameter_name : link list ;
  mutable state_machine : (state_machine_instr * Instr.operand list) list ;
}
```

Dex.debug_info_item[5.1] referenced from Dex.code_item[5.1]

```
type state_machine_instr =
  | DBG_END_SEQUENCE
```

0x00

```
  | DBG_ADVANCE_PC
```

0x01

```
  | DBG_ADVANCE_LINE
```

0x02

```
  | DBG_START_LOCAL
```

0x03

```
  | DBG_START_LOCAL_EXTENDED
```

0x04

```
  | DBG_END_LOCAL
```

0x05

```
  | DBG_RESTART_LOCAL
```

0x06

```
  | DBG_SET_PROLOGUE_END
```

0x07

```
  | DBG_SET_EPILOGUE_BEGIN
```

0x08

```
  | DBG_SET_FILE
```

0x09

```
  | DBG_SPECIAL
```
0x0a..0xff

byte code values for `state_machine` inside `Dex.debug_info_item`[5.1]

```
type anno_dir_item = {
  class_anno_off : link ;
  fields : anno_off list ;
  methods : anno_off list ;
  parameters : anno_off list ;
}
```
`annotations_directory_item` referenced from `Dex.class_def_item`[5.1]

```
type anno_off = {
  target : link ;
  annotation_off : link ;
}
```
`(field|method|parameter)_annotation` format used at `Dex.anno_dir_item`[5.1]

```
type annotation_item = {
  visible : visibility ;
  annotation : encoded_annotation ;
}
```
`Dex.annotation_item`[5.1] referenced from `ANNO_SET`

```
type visibility =
  | VISIBILITY_BUILD
```
0x00

```
  | VISIBILITY_RUNTIME
```
0x01

```
  | VISIBILITY_SYSTEM
```
0x02

`Visibility` values

```
type encoded_annotation = {
  mutable an_type_idx : link ;
  elements : annotation_element list ;
}
```
`Dex.encoded_annotation`[5.1] format referenced from `Dex.encoded_value`[5.1]

```
type annotation_element = {
  name_idx : link ;
  mutable value : encoded_value ;
}
```

`Dex.annotation_element`[5.1] format referenced from `Dex.encoded_annotation`[5.1]

```
type encoded_value =
  | VALUE_BYTE of int64
```
0x00
```
  | VALUE_SHORT of int64
```
0x02
```
  | VALUE_CHAR of int64
```
0x03
```
  | VALUE_INT of int64
```
0x04
```
  | VALUE_LONG of int64
```
0x06
```
  | VALUE_FLOAT of int64
```
0x10
```
  | VALUE_DOUBLE of int64
```
0x11
```
  | VALUE_STRING of int
```
0x17
```
  | VALUE_TYPE of int
```
0x18
```
  | VALUE_FIELD of int
```
0x19
```
  | VALUE_METHOD of int
```
0x1a
```
  | VALUE_ENUM of int
```
0x1b
```
  | VALUE_ARRAY of encoded_value list
```
0x1c
```
  | VALUE_ANNOTATION of encoded_annotation
```
0x1d
```
  | VALUE_NULL
```
0x1e
```
  | VALUE_BOOLEAN of bool
```
0x1f

`Dex.encoded_value`[5.1] encoding embedded in `Dex.annotation_element`[5.1] and `encoded_array`

5.2 Utilities

`val to_idx : int -> link`

wrapping with `Idx`

`val to_off : int -> link`

wrapping with `Off`

`val of_idx : link -> int`

unwrapping `Idx`

`val of_off : link -> int`

unwrapping `Off`

```
module IdxKey :
  sig
     type t = Dex.link
     val compare : t -> t -> int
  end
```

```
module OffKey :
  sig
     type t = Dex.link
     val compare : t -> t -> int
  end
```

`val opr2idx : Instr.operand -> link`

from `OPR_INDEX` to `Idx`

`val opr2off : Instr.operand -> link`

from `OPR_OFFSET` to `Off`

`val idx2opr : link -> Instr.operand`

from `Idx` to `OPR_INDEX`

`val off2opr : link -> Instr.operand`

from `Off` to `OPR_OFFSET`

`val get_off : link -> Instr.offset`

obtain 32-bits offset from `Off`

`val str_to_endian : string -> endian`

obtain `Dex.endian`[5.1] from `string` representation `val endian_to_str : endian -> string`

  `string` representation of `Dex.endian`[5.1]

`val to_type_code : int -> type_code`

  convert `int` to corresponding `Dex.type_code`[5.1]

`val of_type_code : type_code -> int`

  get `int` value of given `Dex.type_code`[5.1]

`val type_code_to_str : type_code -> string`

  get `string` notation of given `Dex.type_code`[5.1]

`val machine_instr_to_str : state_machine_instr -> string`

  get `string` notation of given `Dex.state_machine_instr`[5.1]

5.3 Access flags

```
type access_flag =
  | ACC_PUBLIC
```
0x1, for all kinds

```
  | ACC_PRIVATE
```
0x2, for all kinds

```
  | ACC_PROTECTED
```
0x4, for all kinds

```
  | ACC_STATIC
```
0x8, for all kinds

```
  | ACC_FINAL
```
0x10, for all kinds

```
  | ACC_SYNCHRONIZED
```
0x20, only for methods

```
  | ACC_VOLATILE
```
0x40, only for fields

```
  | ACC_BRIDGE
```
0x40, only for methods

```
  | ACC_TRANSIENT
```
0x80, only for fields

```
  | ACC_VARARGS
```
0x80, only for methods

`| ACC_NATIVE`
: 0x100, only for methods

`| ACC_INTERFACE`
: 0x200, only for classes

`| ACC_ABSTRACT`
: 0x400, except for fields

`| ACC_STRICT`
: 0x800, only for methods

`| ACC_SYNTHETIC`
: 0x1000, for all kinds

`| ACC_ANNOTATION`
: 0x2000, only for classes

`| ACC_ENUM`
: 0x4000, except for methods

`| ACC_CONSTRUCTOR`
: 0x10000, only for methods

`| ACC_DECLARED_SYNCHRONIZED`
: 0x20000, only for methods

indicate the accessibility

```
type acc_kind =
  | ACC_FOR_CLASSES
  | ACC_FOR_FIELDS
  | ACC_FOR_METHODS
```

distinguish targets for `Dex.access_flag`[5.3]

```
val to_acc_flag : acc_kind -> access_flag list -> int
```

make `int` representation from bitfields of `Dex.access_flag`[5.3]

```
val chk_acc_flag : acc_kind -> access_flag list -> int -> bool
```

check certain flags are set

```
val is_static : int -> bool
```

`true` if `ACC_STATIC` is set

```
val is_interface : int -> bool
```

`true` if `ACC_INTERFACE` is set

```
val is_synthetic : int -> bool
```

`true` if `ACC_SYNTHETIC` is set

`val pub : access_flag list`
    `ACC_FOR_PUBLIC`

`val spub : access_flag list`
    `ACC_STATIC` along with `Dex.pub`[5.3]

5.4 Navigation

`val no_index : int`
    0xffffffff (= -1 if signed int)

`val no_offset : int`
    0x00000000

`val no_idx : link`
    wrapping `Dex.no_index`[5.4] with `Idx`

`val no_off : link`
    wrapping `Dex.no_offset`[5.4] with `Off`

`val get_data_item : dex -> link -> data_item`
    get `Dex.data_item`[5.1] for given offset

`val get_ins : dex -> link -> Instr.instr`
    get `Instr.instr`[4.1] for given offset, raise `Dex.Wrong_match`[5.1] unless `INSTRUCTION`

`val is_ins : dex -> link -> bool`
    `true` if the item for given offset is `Instr.instr`[4.1]

`val get_str : dex -> link -> string`
    get `string` for given string id, raise `Dex.Wrong_match`[5.1] unless `STRING_DATA`

`val find_str : dex -> string -> link`
    find string id for given `string`, `Dex.no_idx`[5.4] unless found

`val get_ty_str : dex -> link -> string`
    get type name for given type id

`val find_ty_str : dex -> string -> link`
    find type id for given `string`, `Dex.no_idx`[5.4] unless found

`val ty_comp : dex -> link -> link -> int`
    comparator for type ids

`val get_ty_lst : dex -> link -> link list`
: get `TYPE_LIST` for given offset, raise `Dex.Wrong_match`[5.1] unless `TYPE_LIST`

`val get_fit : dex -> link -> field_id_item`
: get `Dex.field_id_item`[5.1] for given field id

`val get_mit : dex -> link -> method_id_item`
: get `Dex.method_id_item`[5.1] for given method id

`val get_pit : dex -> method_id_item -> proto_id_item`
: get `Dex.proto_id_item`[5.1] for a given method.

`val get_argv : dex -> method_id_item -> link list`
: get a `list` of arguments for given method

`val get_rety : dex -> method_id_item -> link`
: get return type for given method

`val fld_comp : dex -> field_id_item -> field_id_item -> int`
: comparator for field signatures: field name and type

`val ty_lst_comp : dex -> link list -> link list -> int`
: comparator for a `list` of type ids

`val ty_lst_comp_relaxed : dex -> link list -> link list -> int`
: comparator for a `list` of type ids, but ignore the package name for types.

`val mtd_comp : dex -> method_id_item -> method_id_item -> int`
: comparator for method signatures: method name, return type, and arguments

`val mtd_comp_relaxed : dex -> method_id_item -> method_id_item -> int`
: comparator for method signatures: method name, return type, and arguments, but ignore the package name for return types and arguments.

`val get_cid_from_fid : dex -> link -> link`
: get class id from field id

`val get_cid_from_mid : dex -> link -> link`
: get class id from method id

`val get_fld_name : dex -> link -> string`
: get name for given field

`val get_mtd_name : dex -> link -> string`
: get name for given method

`val get_fld_full_name : dex -> link -> string`

get name for given field, along with class name

`val get_mtd_full_name : dex -> link -> string`

get name for given method, along with class name

`val get_mtd_sig : dex -> link -> string`

get method signature, e.g., `Lpkg/cls;->mtd(arg1;arg2;...)rety`

`val get_cid : dex -> string -> link`

get class id from name, `Dex.no_idx`[5.4] unless found

`val get_cdef : dex -> link -> class_def_item`

get `Dex.class_def_item`[5.1] for given class id, raise `Not_found` unless found

`val get_interfaces : dex -> link -> link list`

get interfaces implemented by the given class

`val get_implementers : dex -> link -> link list`

get classes that implement the given interface

`val get_superclass : dex -> link -> link`

get super class id for given class, `Dex.no_idx`[5.4] if it's at the top level

`val get_superclasses : dex -> link -> link list`

get super classes for a given class

`val in_hierarchy : dex -> (link -> bool) -> link -> bool`

check that some property (given as a function `Dex.link`[5.1] to `bool`) holds in hierarchy starting from the given class

`val is_superclass : dex -> link -> link -> bool`

check whether some class is a super class (up through the hierarchy) of a given class

`val is_innerclass : dex -> link -> link -> bool`

check whether some class is an inner class of the given class

`val get_innerclasses : dex -> link -> link list`

get inner classes for the given class

`val get_owning_class : dex -> link -> link`

get owning class if the given class is an inner class

`val get_flds : dex -> link -> (link * field_id_item) list`

get all fields, along with ids, for given class `val get_fldS : dex -> link -> (link * field_id_item) list`

get all fields, along with ids, for given class and superclasses

`val get_the_fld : dex -> link -> string -> link * field_id_item`

get the specific field of given class and given field name

`val own_the_fld : dex -> link -> link -> bool`

`true` if the class owns the field

`val get_mtds : dex -> link -> (link * method_id_item) list`

get all methods, along with ids, for given class

`val get_mtdS : dex -> link -> (link * method_id_item) list`

get all methods, along with ids, for given class and superclasses

`val get_supermethod : dex -> link -> link -> link`

get overriden method at the superclass for given class and method, `Dex.no_idx`[5.4] unless overridable

`val get_the_mtd : dex -> link -> string -> link * method_id_item`

get the specific method of given class and given method name

```
val get_the_mtd_shorty :
  dex -> link -> string -> string -> link * method_id_item
```

get the specific method of given class, method name, and shorty descriptor (useful for overloading)

`val own_the_mtd : dex -> link -> link -> bool`

`true` if the class owns the method

`val get_cdata : dex -> link -> link * class_data_item`

get `Dex.class_data_item`[5.1] for given class, raise `Dex.Wrong_match`[5.1] unless `CLASS_DATA`

`val get_stt_flds : dex -> link -> (link * encoded_value option) list`

get static fields for given class, along with initial values if exists

`val get_emtd : dex -> link -> link -> encoded_method`

get `Dex.encoded_method`[5.1] for given class and method, raise `Dex.Wrong_dex`[5.1] if such method is not defined

`val get_citm : dex -> link -> link -> link * code_item`

get `Dex.code_item`[5.1] for given class and method, raise `Dex.Wrong_match`[5.1] unless `CODE_ITEM`

`val calc_this : code_item -> int`

calculate a register number that holds `this` pointer

`val is_param : code_item -> int -> bool`
: `true` if the given register is used as a parameter

5.5 Modification helper

`val empty_section : unit -> section`
: empty `Dex.section`[5.1]

`val empty_dex : unit -> dex`
: empty `Dex.dex`[5.1]

`val empty_citm : unit -> code_item`
: empty `Dex.code_item`[5.1]

`val insrt_data : dex -> link -> data_item -> unit`
: insert `Dex.data_item`[5.1] into the data pool

`val rm_data : dex -> link -> unit`
: remove `Dex.data_item`[5.1] in the data pool

`val insrt_ins : dex -> link -> Instr.instr -> unit`
: insert `Instr.instr`[4.1] into the data pool

`val insrt_str : dex -> link -> string -> unit`
: insert `string` into the data pool

`val insrt_ty_lst : dex -> link -> link list -> unit`
: insert `TYPE_LIST` into the data pool

`val insrt_stt : dex -> link -> encoded_value list -> unit`
: insert `STATIC_VALUE` into the data pool

`val insrt_citm : dex -> link -> code_item -> unit`
: insert `Dex.code_item`[5.1] into the data pool

6 Module `Parse` : This module provides a function for parsing binary input channel.

`val parse : Pervasives.in_channel -> Dex.dex`
: parse DEX binary `in_channel` into `Dex.dex`[5.1]

7 Module `Visitor` : This module provides visitor pattern.

```
class type visitor =
  object
```

`val dx : Dex.dex`

`method get_dx : unit -> Dex.dex`

invoke if you want to get modified `Dex.dex`[5.1]

`method v_fit : Dex.field_id_item -> unit`

visiting `Dex.field_id_item`[5.1]

`method v_mit : Dex.method_id_item -> unit`

visiting `Dex.method_id_item`[5.1]

`val mutable skip_cls : bool`

skip the current class

`method get_skip_cls : unit -> bool`

getter for `Visitor.visitor.skip_cls`[7]

`method v_cdef : Dex.class_def_item -> unit`

visiting `Dex.class_def_item`[5.1]

`method r_eval : Dex.encoded_value -> Dex.encoded_value`

remapping `Dex.encoded_value`[5.1] stored at `STATIC_VALUE`

`method v_anno : Dex.encoded_annotation -> unit`

visiting `Dex.encoded_annotation`[5.1] stored at `ANNOTATION`

`method v_cdat : Dex.class_data_item -> unit`

visiting `Dex.class_data_item`[5.1]

`method v_efld : Dex.encoded_field -> unit`

visiting `Dex.encoded_field`[5.1]

`val mutable skip_mtd : bool`

skip the current method

`method get_skip_mtd : unit -> bool`

getter for `Visitor.visitor.skip_mtd`[7]

`method v_emtd : Dex.encoded_method -> unit`

visiting `Dex.encoded_method`[5.1]

`method v_citm : Dex.code_item -> unit`

visiting `Dex.code_item`[5.1]

`method v_ins : Dex.link -> unit`

visiting `INSTRUCTION`

`method v_try : Dex.try_item -> unit`

visiting `Dex.try_item`[5.1]

`method v_hdl : Dex.encoded_catch_handler -> unit`

visiting `Dex.encoded_catch_handler`[5.1]

`method v_dbg : Dex.debug_info_item -> unit`

visiting `Dex.debug_info_item`[5.1]

`method finish : unit -> unit`

invoked after traversing `Dex.dex`[5.1]

`end`

visitor

`class iterator : Dex.dex -> visitor`

iterator

`val set_skip_pkgs : string list -> unit`

set packages names to be skipped

`val to_be_skipped : string -> bool`

`true` if the given class name is set to be skipped

`val iter : visitor -> unit`

traversing `Dex.dex`[5.1] using `Visitor.iterator`[7]-like `Visitor.visitor`[7]

8 Module `Android` : This module provides utility functions for handling `Android` platform, for example, it provides a number of typical names for commonly used classes within the Android framework (helpful for finding and manipulating things like Buttons, for example)

8.1 Libraries

```
module App :
  sig

     val activity : string

         android.app.Activity

     val service : string

         android.app.Service

     val application : string

         android.app.Application

     val lst_act : string

         android.app.ListActivity

     val tab_act : string

         android.app.TabActivity

     val onCreate : string

         onCreate

     val onStart : string

         onStart

     val onResume : string

         onResume

     val onPause : string

         onPause

     val onStop : string

         onStop
```

```
    val onDestroy : string
        onDestroy
    val onBind : string
        onBind
    val onRebind : string
        onRebind
    val onUnbind : string
        onUnbind
    val onCreateOptionsMenu : string
        onCreateOptionsMenu
    val onOptionsItemSelected : string
        onOptionsItemSelected
    val set_view : string
        setContentView
    val find_view : string
        findViewById
    val query : string
        managedQuery
    val lifecycle_act : string list
        Activity lifecycle methods
    val lifecycle_srv : string list
        Service lifecycle methods
  end
module Content :
  sig
    val context : string
        android.content.Context
    val intent : string
        android.content.Intent

    val provider : string
        android.content.ContentProvider

    val uris : string
        android.content.ContentUris

    val pwr_serv : string
        android.content.Context.POWER_SERVICE

    val loc_serv : string
        android.content.Context.LOCATION_SERVICE

    val con_serv : string
        android.content.Context.CONNECTIVITY_SERVICE

    val get_sys_serv : string
        getSystemService

    val chk_perm : string
        checkCallingOrSelfPermission

    val appended : string
        withAppendedId

    val query : string
        query

    val set_class : string
        setClass

    val start_act : string
        startActivity

    val start_srv : string
        startService

    val uri : string
        content://com.android.contacts
    module PM :
      sig
        val pm : string
            android.content.pm.PackageManager
        val chk_perm : string
            checkPermission
      end
  end
module Database :
  sig
     val cursor : string
         android.database.Cursor
     val get_col_idx : string
         getColumnIndex
     val get_col_idx_e : string
         getColumnIndexOrThrow
  end
module Location :
  sig
     val manager : string
         android.location.LocationManager
  end
module Net :
  sig
     val uri : string
         android.net.Uri
     val parse : string
         parse
     val getHost : string
         getHost
     val appended : string

         withAppendedPath

  end

module OS :
  sig

     val bundle : string

         android.os.Bundle

     val iitf : string

         android.os.IInterface

  end

module Preference :
  sig

     val activity : string

         android.preference.PreferenceActivity

  end

module Util :
  sig

     val log : string

         android.util.Log

  end

module View :
  sig

     val key : string

         android.view.KeyEvent

     module KeyEvent :
       sig

         val onKeyDown : string
            onKeyDown

         val onKeyLongPress : string
            onKeyLongPress
```

```
      val onKeyMultiple : string
          onKeyMultiple

      val onKeyUp : string
          onKeyUp

      val is_key_abstract : string -> bool
          true if given method is one of KeyEvent abstracts

    end

  val menu : string

      android.view.MenuItem

  module MenuItem :
    sig

      val onMenuItemClick : string
          onMenuItemClick

      val is_menu_abstract : string -> bool
          true if given method is one of MenuItem abstracts

    end

  val view : string

      android.view.View

  val onClick : string

      onClick

  val onKey : string

      onKey

  val onLongClick : string

      onLongClick

  val onTouch : string

      onTouch

  val is_view_abstract : string -> bool

      true if given method is one of View abstracts

end
```

8.2 Utilities

`val is_library : string -> bool`

  `true` if given class is `Android` library

`val is_static_library : string -> bool`

  `true` if given class is `Android` static library

`val is_abstract : string -> bool`

  `true` if given method is abstract

`val is_context : Dex.dex -> Dex.link -> bool`

  `true` if given class is subclass of `Context`

`val is_activity : Dex.dex -> Dex.link -> bool`

  `true` if given class is subclass of `Activity`

`val is_fragment : Dex.dex -> Dex.link -> bool`

  `true` if given class is subclass of `Fragment`

`val is_listener : Dex.dex -> Dex.link -> bool`

  `true` if given class implements any sorts of `Listener`

`val find_lifecycle_act : Dex.dex -> Dex.link -> Dex.link list`

  find `Activity` lifecycle methods

`val is_set_listener : Dex.dex -> Dex.link -> bool`

  `true` if given method is a setter for `Listener`

8.3 Permissions

```
module Permission :
  sig
```

  `val internet : string`

    `android.permission.INTERNET`

  `val read_contacts : string`

    `android.permission.READ_CONTACTS`

  `val access_fine_location : string`

    `android.permission.ACCESS_FINE_LOCATION`

  `val access_coarse_location : string`

```
        android.permission.ACCESS_COARSE_LOCATION

    val read_phone_state : string

        android.permission.READ_PHONE_STATE

    val write_settings : string

        android.permission.WRITE_SETTINGS

    val access_network_state : string

        android.permission.ACCESS_NETWORK_STATE

    val change_network_state : string

        android.permission.CHANGE_NETWORK_STATE

  end
```

8.4 Ads

```
module Ads :
  sig
    val is_ads_pkg : string -> bool
```

        `true` if given class is inside `Ads` package

```
  end
```

8.5 Analyses

`val sdk : string Pervasives.ref`
SDK name of interest

`val api_usage : Dex.dex -> unit`
report API usage (including overrides) in the dex file

9 Module `Unparse` : This module provides utility functions for pretty printing or collecting information about DEX file.

9.1 Pretty Printing

`val unparse : Dex.dex -> unit`
print `Dex.dex`[5.1] in YAML format

`val print_method : Dex.dex -> Dex.code_item -> unit`
print `Dex.code_item`[5.1] as a method

9.2 Collecting Information

`val print_info : Dex.dex -> unit`

print basic infomation about DEX file

`val print_classes : Dex.dex -> unit`

print all the class names occurred in DEX file

10 Module `Htmlunparse` : This module provides functions for dumping contents of dex files in a directory based html structure, allowing the viewer to jump around the directory to more easily visualize code.

`val generate_documentation : Dex.dex -> string -> string -> unit`

10.1 Generate HTML output

11 Module `Callgraph` : This module defines a type for call graph and provides functions for generating and printing a call graph.

`type cg`

Call Graph

`val add_call : Dex.dex -> cg -> Dex.link -> Dex.link -> bool`

into the `Callgraph.cg`[11], add an edge from the caller to the callee returns `true` if a new node is introduced

`val make_cg : Dex.dex -> cg`

make call graph for overall `Dex.dex`[5.1] file

`val make_partial_cg : Dex.dex -> int -> Dex.link list -> cg`

partial call graph starting from the given classes, with a certain depth

`type cc = Dex.link list`

Call Chain in a reversed order

`val compare_cc : cc -> cc -> int`

call chain comparison

`val callers : Dex.dex -> int -> cg -> Dex.link -> cc list`

find callers for the given method, with a certain depth

`val has_caller : Dex.dex -> cg -> Dex.link -> bool`

`true` if the given method is invoked by other methods

`val dependants : Dex.dex -> cg -> Dex.link -> Dex.link list`

find dependent classes for the given class

`val cg2dot : Dex.dex -> cg -> unit`

print `Callgraph.cg`[11] in dot format

12 Module `Ctrlflow` : This module defines types for control-flow graph and dominance relations, and provides utility functions for obtaining information from such graphs.

12.1 Control-Flow Graph

`type cfg`

Control-Flow Graph

`val make_cfg : Dex.dex -> Dex.code_item -> cfg`

make control-flow graph for given `Dex.code_item`[5.1]

12.2 Dominator Tree

`type dom`

Dominator Tree

`val doms : cfg -> dom`

compute block-level dominance relations for given `Ctrlflow.cfg`[12.1]

`val idom : dom -> int -> int`

immediate dominator according to dominance relations

`val cdom : dom -> int list`

longest common dominators

12.3 Post Dominator Tree

`type pdom`

Post Dominator Tree

`val pdoms : cfg -> pdom`

compute block-level post dominace relations for given `Ctrlflow.cfg`[12.1]

`val ipdom : pdom -> int -> int`

immediate post dominator according to post dominace realtions

`val cpdom : pdom -> int list`

longest common post dominators

`val get_last_inss : cfg -> pdom -> Dex.link list`

get the last instructions

12.4 Control-flow Module for Data-flow Analysis

```
module type CTRLFLOW =
  sig
```

`type st`

statement type

`val start : st`

starting statement

`val last : st list`

last statements

`val all : st list`

all statements

`val pred : st -> st list`

predecessors

`val succ : st -> st list`

successors

`val to_s : st -> string`

for debugging

```
  end
```

Control-flow

```
type cfg_module = (module Ctrlflow.CTRLFLOW with type st = Dex.link)
val to_module : Dex.dex -> cfg -> cfg_module
```

make `Ctrlflow.cfg_module`[12.4] type module using `Ctrlflow.cfg`[12.1]

12.5 DOTtify

```
val cfg2dot : Dex.dex -> cfg -> unit
```

print control-flow graph in dot format

```
val dom2dot : Dex.dex -> cfg -> dom -> unit
```

print dominator tree in dot format

```
val pdom2dot : Dex.dex -> cfg -> pdom -> unit
```

print post dominator tree in dot format

13 Module `Dataflow` : This module provides data-flow analysis frameworks

```
module type SCHEDULER =
  sig
```

```
    type st
```

statement type

```
    val hasNext : unit -> bool
```

`true` if it has a next `Dataflow.SCHEDULER.st`[13]

```
    val next : unit -> st
```

return the next available `Dataflow.SCHEDULER.st`[13]

```
    val add : st -> unit
```

add the given `Dataflow.SCHEDULER.st`[13] into the scheduler

```
  end
```

Scheduler

```
module Worklist :
   SCHEDULER  with type st = Dex.link
```

simple queue-based scheduler

```
module type LATTICE =
  sig
```

```
type l
```

element type

```
val bot : l
```

`BOTTOM` of the lattice

```
val top : l
```

`TOP` of the lattice

```
val meet : l -> l -> l
```

meet operator

```
val compare : l -> l -> int
```

partial order between `Dataflow.LATTICE.l`[13]s

```
val to_s : l -> string
```

convert `Dataflow.LATTICE.l`[13] to `string`

```
  end
```

Lattice

```
module type DATAFLOW =
  sig
```

```
type l
```

same as `Dataflow.LATTICE.l`[13]

```
type st
```

same as `Dataflow.SCHEDULER.st`[13]

```
val init : st -> l
```

initial `Dataflow.LATTICE.l`[13]

```
val trans : l -> st -> l
```

transfer function

```
  end
```

Data-flow

```
module type ANALYSIS =
  sig
```

```
type l
```

same as `Dataflow.LATTICE.l`[13]

```
type st
```

same as `Dataflow.SCHEDULER.st`[13]

```
val to_s : l -> string
```

same as `Dataflow.LATTICE.to_s`[13]

```
val inn : st -> l
```

return `IN` for the given `Dataflow.ANALYSIS.st`[13]

```
val out : st -> l
```

return `OUT` for the given `Dataflow.ANALYSIS.st`[13]

```
val fixed_pt : unit -> unit
```

calculate fixed point

```
  end
```

Data-flow analysis

```
module FwDFA :
   functor (SC : SCHEDULER) -> functor (LT : LATTICE) -> functor (CF : Ctrlflow.CTRLFLOW
with type st = SC.st) -> functor (DF : DATAFLOW  with type st = SC.st and type l = LT.l)
-> ANALYSIS  with type st = SC.st and type l = LT.l
```

Forward Data-flow analysis

```
module BwDFA :
   functor (SC : SCHEDULER) -> functor (LT : LATTICE) -> functor (CF : Ctrlflow.CTRLFLOW
with type st = SC.st) -> functor (DF : DATAFLOW  with type st = SC.st and type l = LT.l)
-> ANALYSIS  with type st = SC.st and type l = LT.l
```

Backward Data-flow analysis

14 Module `Liveness` : This module defines liveness analysis using `Dataflow`[13] module.

```
type liveness = (module Dataflow.ANALYSIS with type l = Util.IS.t and type st = Dex.link)
val make_dfa : Dex.dex -> Dex.code_item -> liveness
```

make liveness analysis

15 Module `Propagation` : This module offers constant propagation analysis using `Dataflow`[13]

```
type value =
  | Const of int64
```

numerical constant

```
  | String of string
```

const-string

```
  | Clazz of string
```

const-class

```
  | Object of string
```

instance

```
  | Intent of string
```

Intent for a specific component

```
  | Field of string * string
```

static fields

```
  | BOT
```

non-const

```
  | TOP
```

undefined

```
val val_to_str : value -> string
```

make `Propagation.value`[15] printable

```
type propagation = (module Dataflow.ANALYSIS with type l = value Util.IM.t and type st =
   Dex.link)
val make_dfa : Dex.dex -> Dex.code_item -> propagation
```

make constant propagation analysis

16 Module Reaching : This module conducts reaching definition analysis via Dataflow[13] module.

```
type reaching = (module Dataflow.ANALYSIS with type l = Dex.link Util.IM.t and type st =
   Dex.link)
```

```
val make_dfa : Dex.dex -> Dex.code_item -> reaching
```

make reaching definition analysis

17 Module Modify : This module provides utility functions for modifying DEX binary.

17.1 Utilities

```
val seed_addr : int -> unit
```

set the start address for fresh ones

17.2 Modification

```
val new_str : Dex.dex -> string -> Dex.link
```

add a new `string`

```
val replace_str : Dex.dex -> string -> string -> bool
```

replace old `string` with new one; `true` if replaced, `false` if newly added

```
val report_str_repl_cnt : unit -> unit
```

report `string` replacement counts

```
val new_ty : Dex.dex -> string -> Dex.link
```

add a new type

```
val new_class :
  Dex.dex -> ?super:string -> string -> Dex.access_flag list -> Dex.link
```

add a new class definition; pass superclass name, its name, and `Dex.access_flag`[5.3]s

```
val make_class_overridable : Dex.dex -> Dex.link -> unit
```

rip off the `final` qualifier on `Dex.class_def_item`[5.1]

```
val add_interface : Dex.dex -> Dex.link -> string -> unit
```

add an interface to a class.

```
val new_field :
  Dex.dex -> Dex.link -> string -> Dex.access_flag list -> string -> Dex.link
```
add a new field definition; pass class id, its name, `Dex.access_flag`[5.3]s, and type

```
val new_sig :
  Dex.dex -> Dex.link -> string -> string -> string list -> Dex.link
```
add a new method signature; pass class id, its name, return type, and arguments

```
val new_method :
  Dex.dex ->
  Dex.link ->
  string -> Dex.access_flag list -> string -> string list -> Dex.link
```
add a new method definition, along with empty body; pass class id, its name, `Dex.access_flag`[5.3]s, return type, and arguments

```
val make_method_overridable : Dex.dex -> Dex.link -> Dex.link -> unit
```
rip off the `final` qualifer on `Dex.encoded_method`[5.1]

```
type cursor
```
instruction inserting point

```
val prev : cursor -> cursor
```
previous instruction

```
val next : cursor -> cursor
```
next instruction

```
val get_cursor : Dex.code_item -> Dex.link -> cursor
```
get the `Modify.cursor`[17.2] of the given instruction

```
val get_fst_cursor : unit -> cursor
```
get the first `Modify.cursor`[17.2]

```
val get_last_cursor : Dex.dex -> Dex.code_item -> cursor
```
get the last `Modify.cursor`[17.2]

```
val get_ins : Dex.dex -> Dex.code_item -> cursor -> Instr.instr
```
get the `Instr.instr`[4.1] at `Modify.cursor`[17.2] point

```
val get_fst_ins : Dex.dex -> Dex.code_item -> Instr.instr
```
get the first `Instr.instr`[4.1]

```
val get_last_ins : Dex.dex -> Dex.code_item -> Instr.instr
```
get the last `Instr.instr`[4.1]

```
val insrt_ins : Dex.dex -> Dex.code_item -> cursor -> Instr.instr -> cursor
```

insert an `Instr.instr`[4.1] at `Modify.cursor`[17.2] point; `Modify.cursor`[17.2] will be advanced

```
val rm_ins : Dex.dex -> Dex.code_item -> cursor -> cursor
```

remove an `Instr.instr`[4.1] at `Modify.cursor`[17.2] point; `Modify.cursor`[17.2] will remain as same

```
val insrt_insns :
  Dex.dex ->
  Dex.code_item -> cursor -> Instr.instr list -> cursor
```

insert `Instr.instr`[4.1]s at `Modify.cursor`[17.2] point; `Modify.cursor`[17.2] will be advanced

```
val insrt_insns_under_off :
  Dex.dex ->
  Dex.code_item -> cursor -> Instr.instr list -> cursor
```

insert `Instr.instr`[4.1]s under the specified `Dex.offset`, while preserving that `Dex.offset` at `Modify.cursor`[17.2] point

```
val insrt_insns_over_off :
  Dex.dex ->
  Dex.code_item -> cursor -> Instr.instr list -> cursor
```

insert `Instr.instr`[4.1]s over the specified `Dex.offset`, while preserving that `Dex.offset` at `Modify.cursor`[17.2] point

```
val insrt_insns_before_start :
  Dex.dex -> Dex.code_item -> Instr.instr list -> cursor
```

insert `Instr.instr`[4.1]s before the start of `Dex.code_item`[5.1]

```
val insrt_insns_after_start :
  Dex.dex -> Dex.code_item -> Instr.instr list -> cursor
```

insert `Instr.instr`[4.1]s after the start of `Dex.code_item`[5.1]

```
val insrt_insns_before_end :
  Dex.dex -> Dex.code_item -> Instr.instr list -> cursor
```

insert `Instr.instr`[4.1]s before the end of `Dex.code_item`[5.1]

```
val insrt_insns_after_end :
  Dex.dex -> Dex.code_item -> Instr.instr list -> cursor
```

insert `Instr.instr`[4.1]s after the end of `Dex.code_item`[5.1]

```
val insrt_return_void : Dex.dex -> Dex.link -> string -> unit
```

insert `Instr.rv`[4.4] at the end of the method

`val shift_reg_usage : Dex.dex -> Dex.code_item -> int -> unit`
shift register usage so as to secure free registers around 0 to avoid register truncations, you may need to call `expand_opr`

`val shift_params : Dex.dex -> Dex.code_item -> int -> unit`
shift parameters so as to secure free registers around "this" to avoid register truncations, you may need to call `expand_opr`

`val update_reg_usage : Dex.dex -> Dex.code_item -> unit`
update register usage: `registers_size` and `outs_size`

`val implements : Dex.dex -> Dex.link -> Dex.link -> string -> bool`
`true`, adding an abstract method if given class doesn't implement it

`val override : Dex.dex -> Dex.link -> string -> bool`
`true`, adding an overriding method if given class doesn't override it

17.3 Application

`val subst_cls : Dex.dex -> string list -> string list -> unit`
substitute the given class usage into the new one

`val rename_cls : Dex.dex -> string list -> unit`
rename specific classes

`val discard_cls_calls : Dex.dex -> string list -> unit`
discard calls related to specific classes

`val call_trace : Dex.dex -> string list -> unit`
trace call stack by modifying methods of specific classes in the dex

`val expand_opr : Dex.dex -> unit`
expand usage caused by massive instrumentations

`val hello : unit -> Dex.dex`
API test

18 Module `Combine` : This module provides a function for merging two DEX binaries.

`val combine : Dex.dex -> Dex.dex -> Dex.dex`
combine two DEX binaries

19 Module `Dump` : This module provides utilities for dumping a dex file into an on disk file.

```
val dump : string -> Dex.dex -> unit
```

dump dex binary for given file name

20 Module `Testing` : instrument testing features into the dex accordingly

```
val modify : Dex.dex -> unit
```

21 Module `Logging` : This module provides special functions for logging apps

```
val add_transition : Dex.dex -> unit
```

add non-overriden transition methods

```
val modify : Dex.dex -> unit
```

instrument logging features into the dex accordingly

22 Module `Main` : Main workhorse

```
val main : unit -> unit
```