PHP を魔改造して学ぶ言語処理系

nsfisis (いまむら)

PHPerKaigi 2026

2026-03-22

いまむら

nsfisis

PHP のソースコードを魔改造して独自の演算子を追加する

PHP のソースコードを魔改造して
独自の演算子を追加する

PHP 処理系の内部実装を理解する

追加する演算子

関数合成演算子「∘」

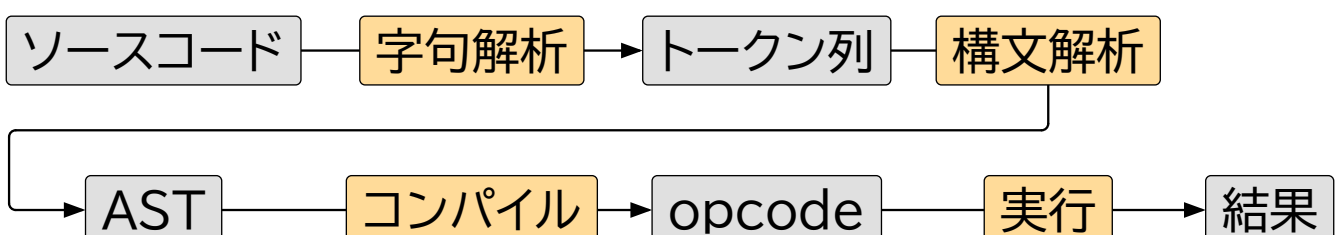
```
$h = $f ∘ $g;
$h($x); // → $f($g($x))
```

$$(f \circ g)(x) = f(g(x))$$

```php
$double = fn($x) => $x * 2;
$add_one = fn($x) => $x + 1;

$f = $add_one ∘ $double;
echo $f(3); // → $add_one($double(3)) → 7
```

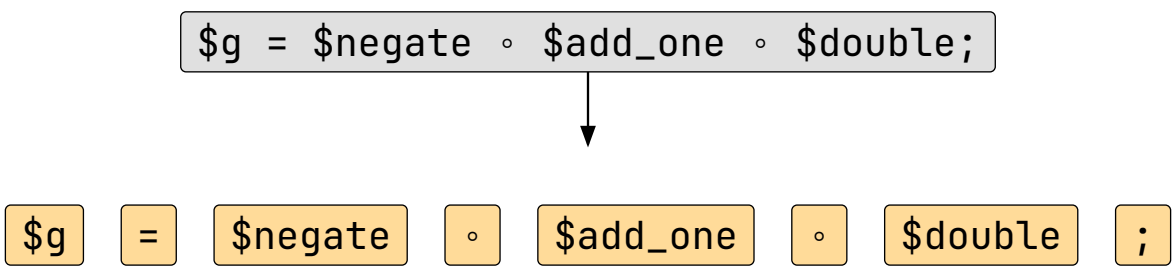
```php
$double = fn($x) => $x * 2;
$add_one = fn($x) => $x + 1;
$negate = fn($x) => -$x;

$g = $negate ∘ $add_one ∘ $double;
echo $g(3);
// → $negate($add_one($double(3)))
// → -7
```

実行の流れ



字句解析

ソースコードの文字列を
トークンの列に分割する処理



re2c

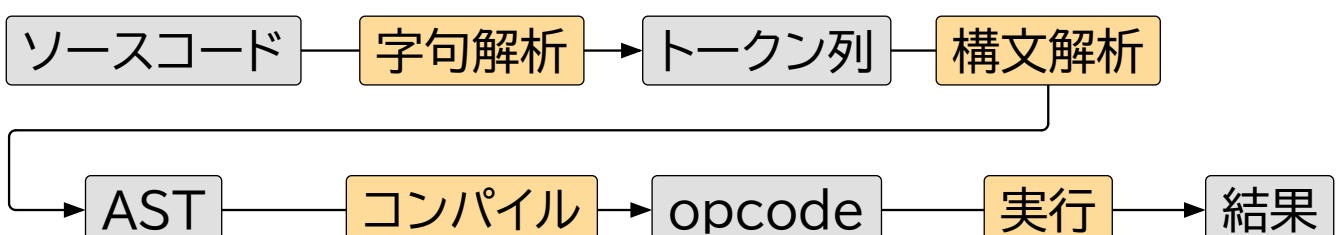
正規表現からコードを生成

zend_language_scanner.l

```
<ST_IN_SCRIPTING>"\xe2\x88\x98" {
    RETURN_TOKEN(T_COMPOSE);
}
```

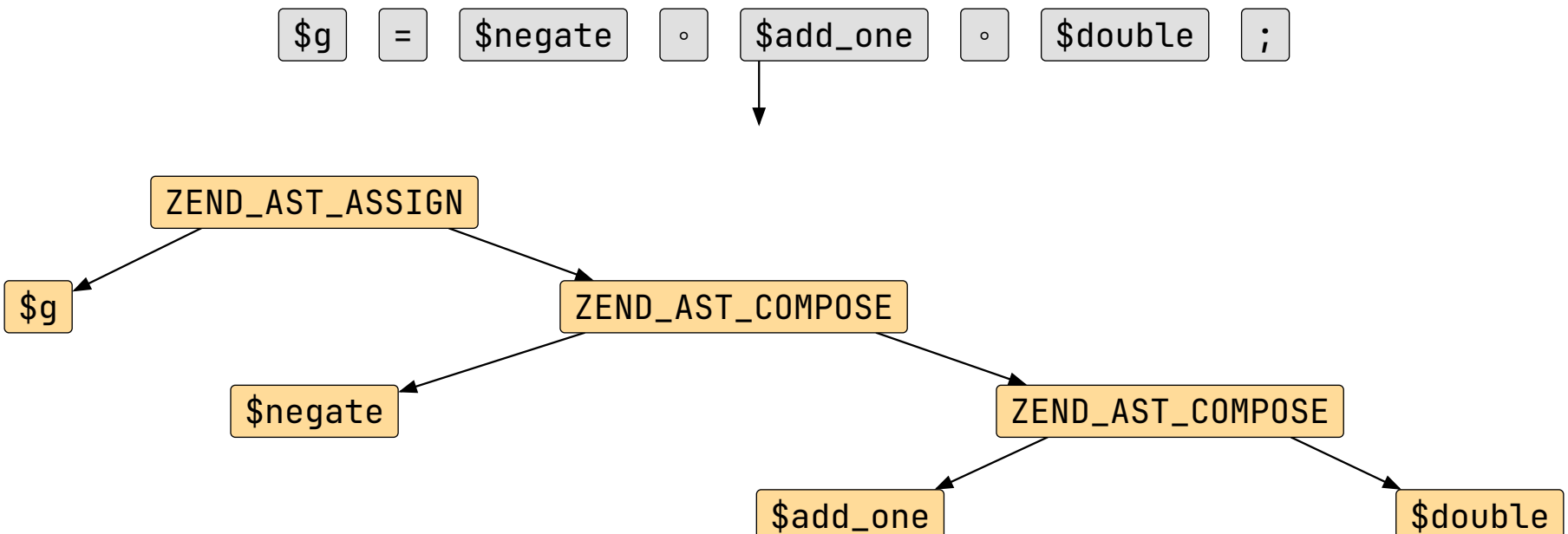
- \xe2\x88\x98 は ∘（U+2218）の UTF-8 バイト列
- T_COMPOSE を返す
- re2c が zend_language_scanner.c を生成

実行の流れ



構文解析

トークン列を
抽象構文木（AST）に変換する処理


$g = $negate ∘ $add_one ∘ $double ;
ZEND_AST_ASSIGN
$g
ZEND_AST_COMPOSE
$negate
ZEND_AST_COMPOSE
$add_one
$double

bison

文法規則からコードを生成

zend_language_parser.y

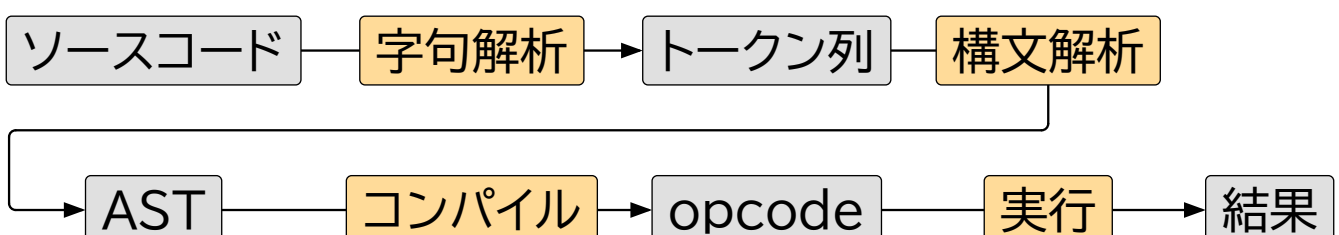
```
/* 優先順位の設定 */
%left T_PIPE
%right T_COMPOSE  /* 追加 */
%left '.'
```

- |> より強い
  - $f ∘ $g |> $h → ($f ∘ $g) |> $h
- 右結合
  - $f ∘ $g ∘ $h → $f ∘ ($g ∘ $h)

zend_language_parser.y

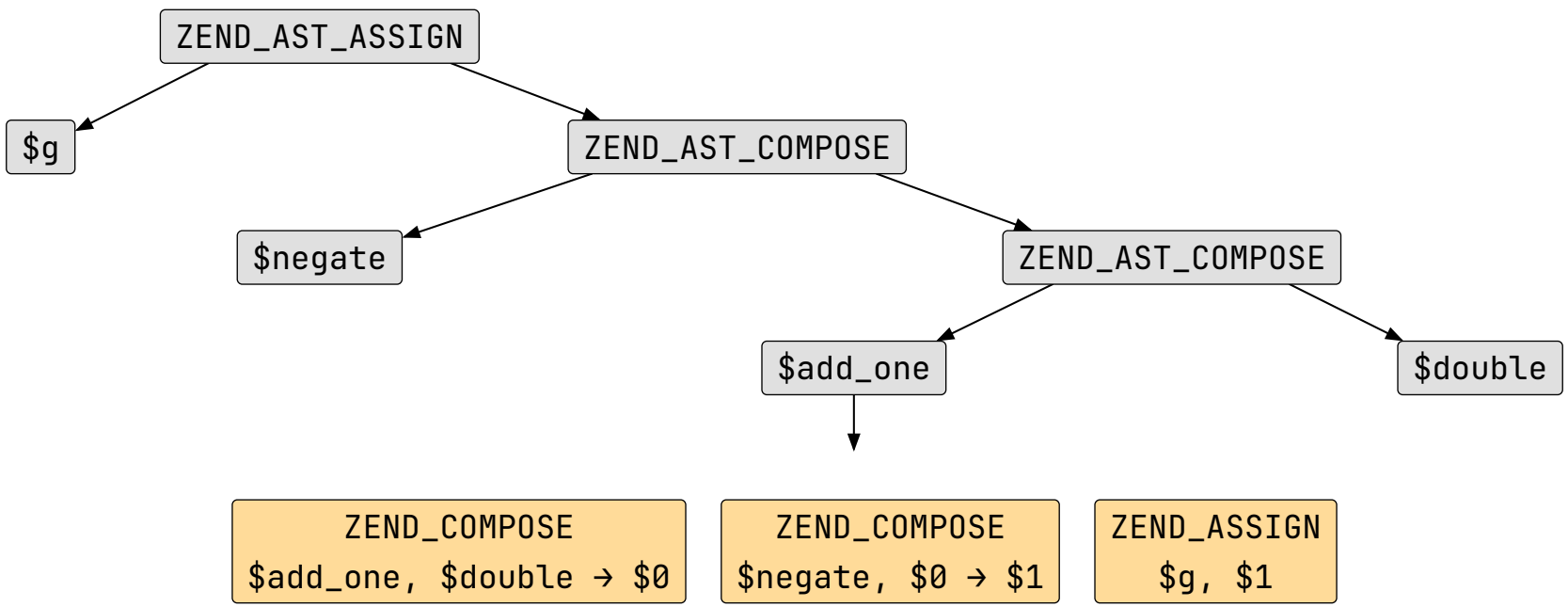
```
/* 文法定義 */
expr : expr T_COMPOSE expr
     {
         /* f ∘ g の並びを見つけたら AST を構築 */
         $$ = zend_ast_create(ZEND_AST_COMPOSE, $1, $3);
     }
```

実行の流れ



コンパイル

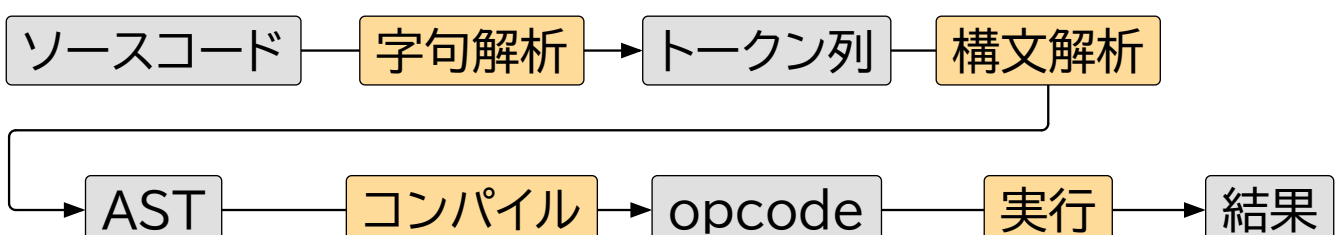
AST を PHP VM の
opcode 列(命令列) に変換する処理


ZEND_AST_ASSIGN
$g
ZEND_AST_COMPOSE
$negate
ZEND_AST_COMPOSE
$add_one
$double
ZEND_COMPOSE
$add_one, $double → $0
ZEND_COMPOSE
$negate, $0 → $1
ZEND_ASSIGN
$g, $1

zend_compile.c

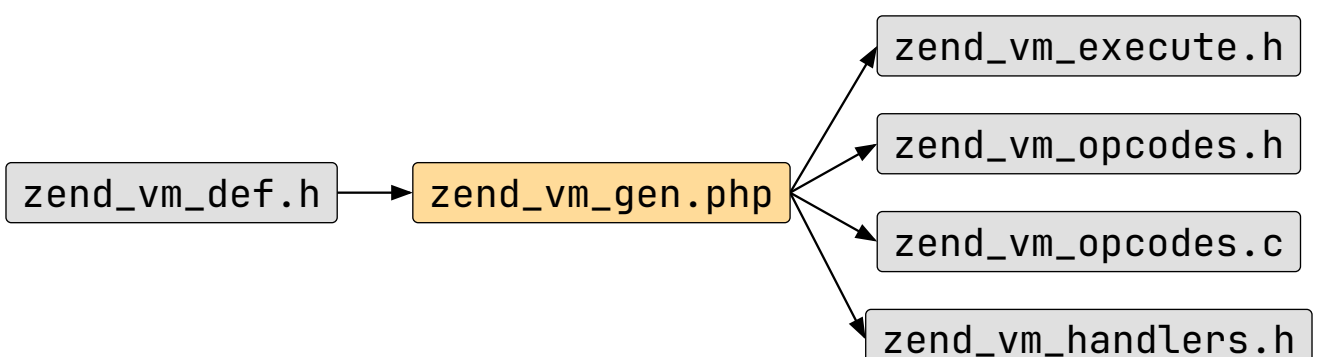
```
static void zend_compile_compose(znode *result, zend_ast *ast) {
    zend_ast *lhs_ast = ast->child[0];
    zend_ast *rhs_ast = ast->child[1];
    znode lhs_result, rhs_result;
    zend_compile_expr(&lhs_result, lhs_ast); // 左辺をコンパイル
    zend_compile_expr(&rhs_result, rhs_ast); // 右辺をコンパイル
    // ZEND_COMPOSE opcode を出力
    // 実際の関数合成は実行時に
    zend_emit_op_tmp(result, ZEND_COMPOSE, &lhs_result, &rhs_result);
}
```

実行の流れ



zend_vm_gen.php

テンプレートから VM コードを生成

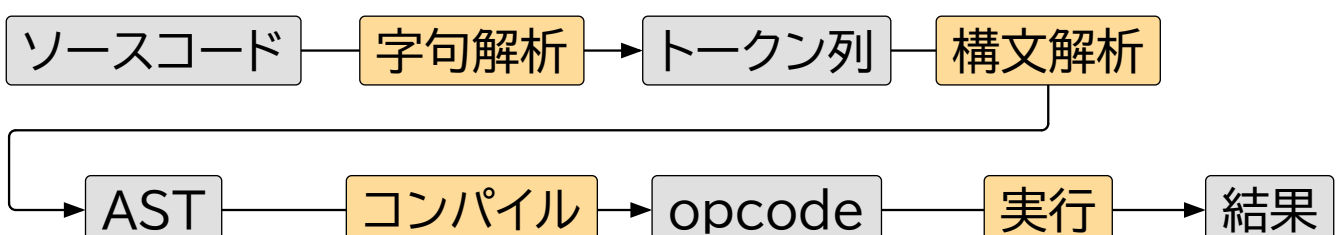


zend_vm_def.h

```
ZEND_VM_HANDLER(
    211,              // opcode の番号
    ZEND_COMPOSE,     // opcode の名前
    CONST|TMPVAR|CV, // operand 1 の種類
    CONST|TMPVAR|CV) // operand 2 の種類
{
    /* (略) */
    zend_compose_callables(EX_VAR(opline->result.var), op1, op2);
    /* (略) */
}
```

```
ZEND_API void zend_compose_callables(zval *result, zval *lhs, zval *rhs)
{
    // (略; いい感じに合成後のクロージャのメタデータを初期化)
    // (略; 左辺と右辺の情報をクロージャの $this に埋め込む)
    // (略; いい感じに結果データを生成)
}
```

実行の流れ



PHPerKaigi 2025

PHP で作る PHP
～セルフホストできる
言語処理系を作ろう～

PHPerKaigi 2026

PHP を魔改造して学ぶ言語処理系

- PHP サブセット実装
  - 複雑なものを単純化して
    全体像を理解
- PHP 魔改造
  - 複雑なものを複雑なまま
    部分的に理解

言語処理系を読もう
言語処理系を書こう

ご静聴
ありがとうございました

参考文献:

- https://docs.raku.org/routine/o,%20infix%20%E2%88%98
- https://docs.julialang.org/en/v1/manual/functions/#Function-composition-and-piping
- https://www.php.net/manual/ja/language.operators.precedence.php
- https://www.npopov.com/2017/04/14/PHP-7-Virtual-machine.html
- https://www.phpinternalsbook.com/