Развитие PHP 7.*

Dmitry Stogov

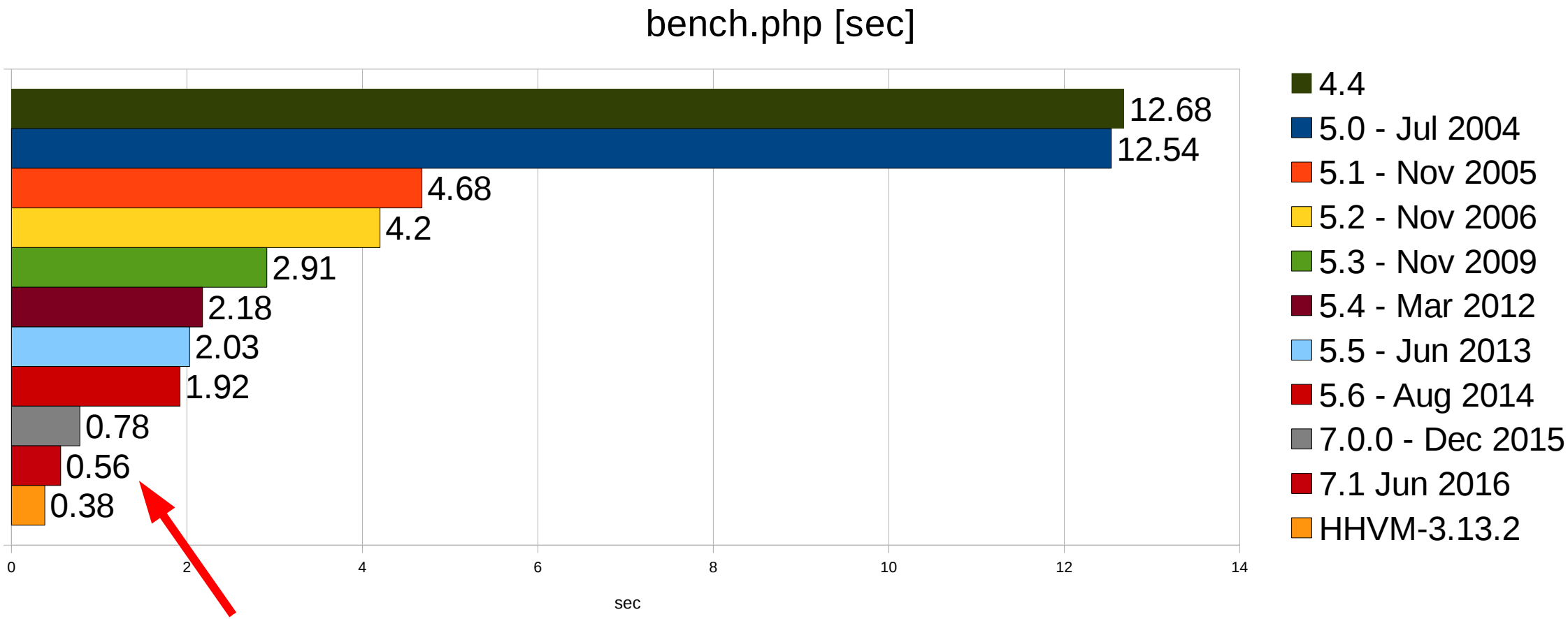
Principal Engineer at Zend Technologies

DevConf 2016, Москва

Кто Я?

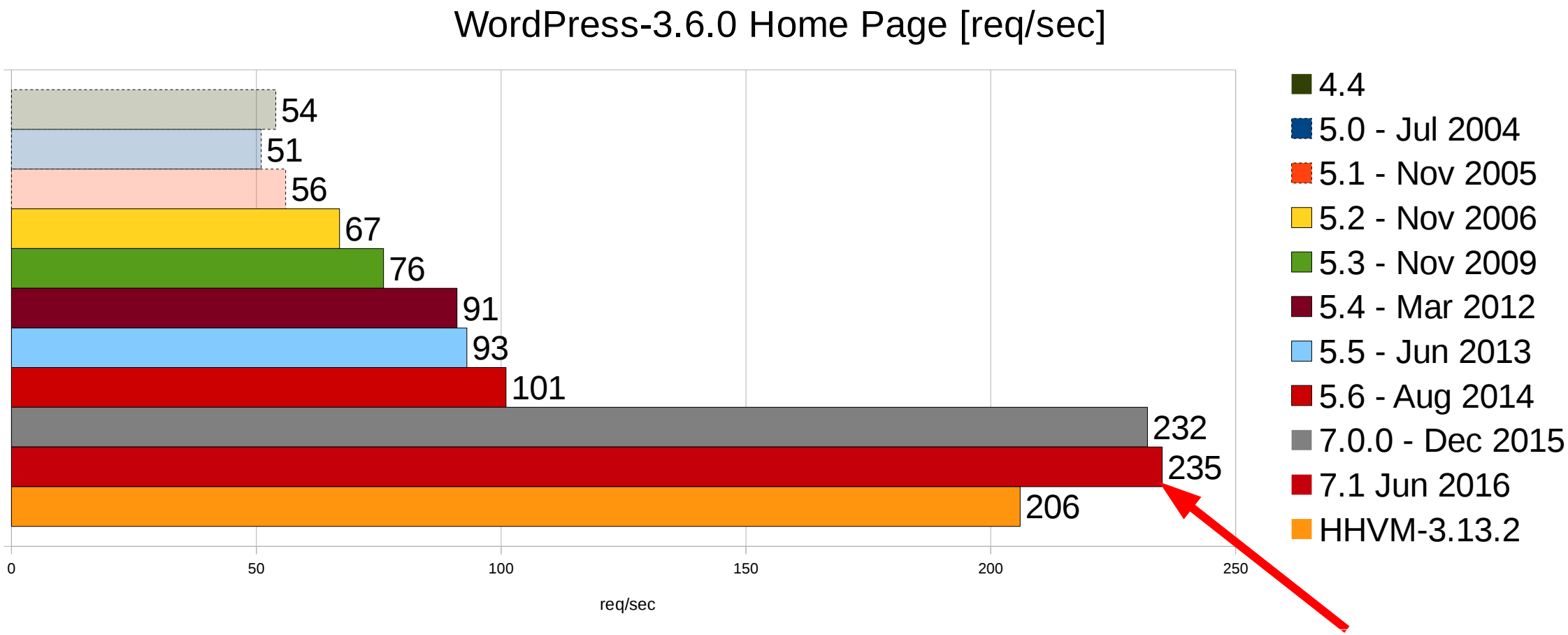
- Работаю в ИТ с 1991
- Первое знакомство с PHP в 2002
- Автор Turck MMCache (eAccelerator)
- Работаю в Zend Technologies с 2004
- Сейчас ведущий инженер
- Автор ext/soap и pecl/perl
- Один из ведущих разработчиков Open Source PHP
- Майнтейнер Zend OPcache
- Лидер проекта PHPNG легшего в основу PHP 7

Производительность PHP на синтетических тестах



PHP-7.1 еще на 25% быстрее PHP-7.0 но все еще медленнее HHVM

Производительность PHP на реальных приложениях



На реальных приложениях PHP-7.1 пока не дает существенного выигрыша!

PHPNG (New Generation)

- Проект получил свое развитие после попыток создания JIT для PHP
- Рефакторинг (ни каких нововведений, 100% совместимость с PHP 5)
- Основная цель — достичь нового уровня производительности и заложить базу для будущих улучшений
- Отделился от основной ветки PHP в январе 2014

- Две недели ушло на то что-бы просто скомпилировать ядро
- Еще через две недели заработал bench.php
- Полтора месяца для обеспечения совместимости с Wordpress
- Еще через месяц (к 9 Мая) мы открыли проект
- В августе 2014 принят как основа для будущего PHP 7

PHP 7.0

- Было решено выпускать PHP 7 после PHP 5, пропустив PHP 6
- GA релиз состоялся в декабре 2015
- Сейчас доступен PHP-7.0.7

- Возможность определять скалярные типы аргументов функций и возвращаемых значений
- Исключения вместо фатальных ошибок
- Анонимный классы
- Zero-cost assert()
- Новые операторы и функции (<=>, ??)
- Чистка неконсистентностей

Badoo перешли на PHP 7.0 и сэкономили $1M

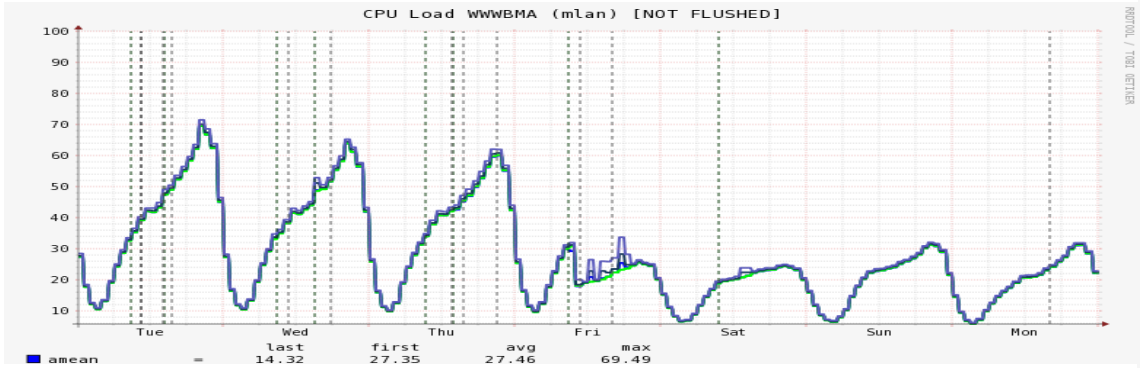
11 марта в 14:22

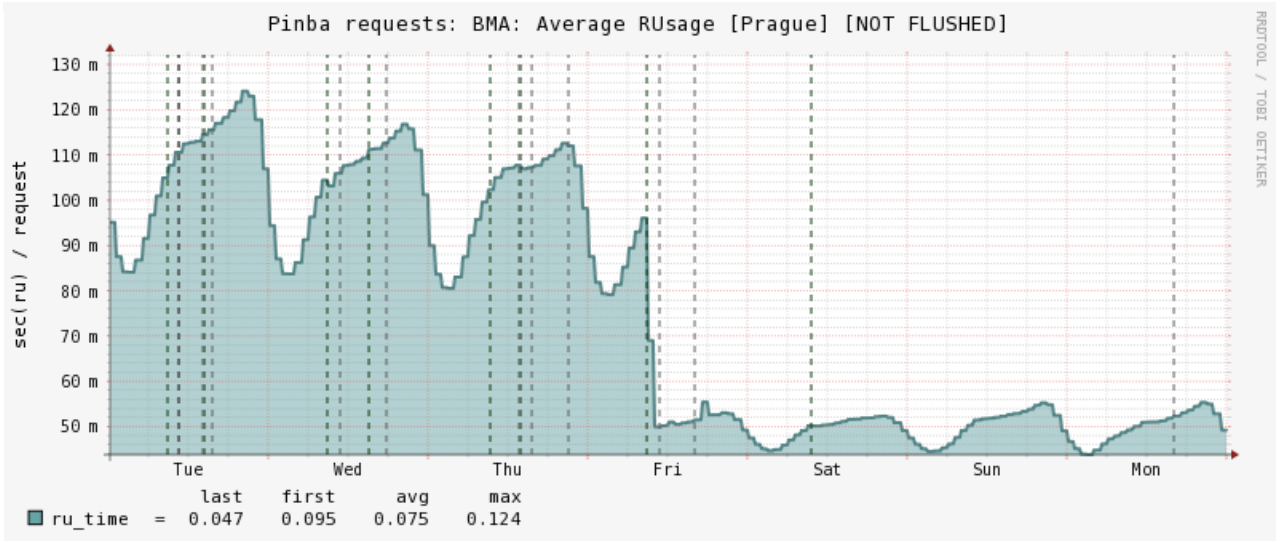
Badoo перешли на PHP7 и сэкономили $1M

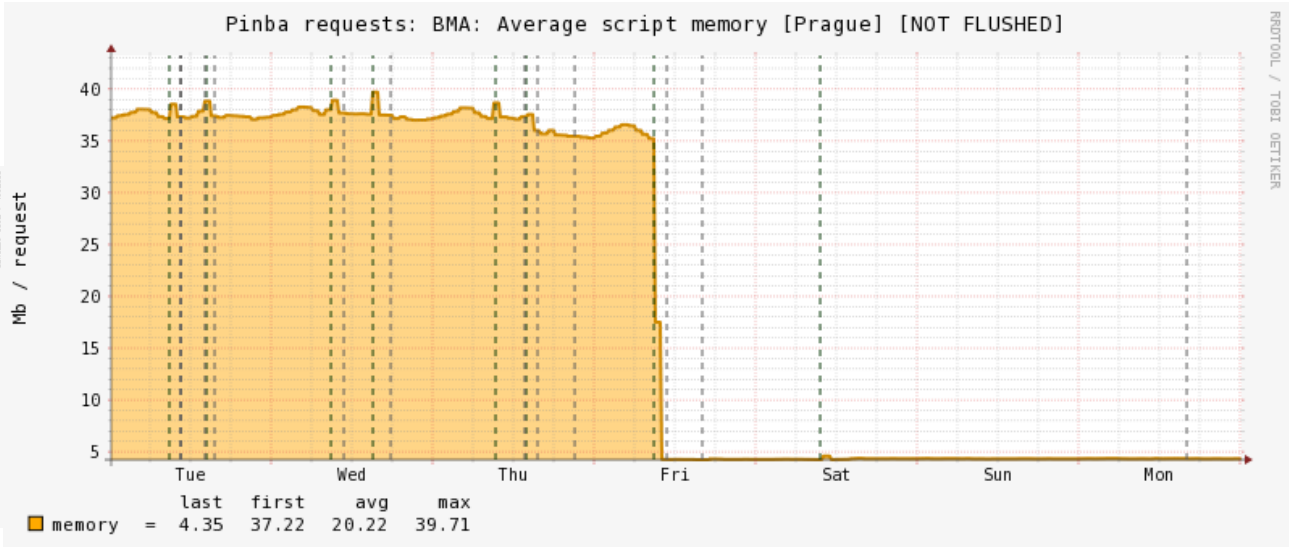
Программирование*, Веб-разработка*, PHP*, Блог компании Badoo

Мы сделали это! Несколько сотен наших application-серверов переведены на PHP7 и прекрасно себя чувствуют. Насколько нам известно, это второй переход на PHP7 проекта такого масштаба (после Etsy). В процессе мы нашли несколько очень неприятных багов в системе кеширования байт-кода PHP7, но они исправлены. А теперь — ура! — благая весть для всего PHP-сообщества: PHP7 действительно готов к продакшену, стабилен, потребляет значительно меньше памяти и дает очень хороший прирост производительности. Ниже мы подробно расскажем, как мы перешли на PHP7, с какими трудностями столкнулись, как с ними боролись и какие результаты получили. Но начнем с небольшого введения.

Мнение о том, что узким местом в веб-проектах является база данных — одно из самых распространенных заблуждений. Хорошо спроектированная система сбалансирована — при увеличении входной нагрузки удар держат все части системы, а при превышении пороговых значений тормозить начинает все: и процессор, и сетевая часть, а не только диски на базах. В этой реальности процессорная мощность application-кластера является чуть ли не самой важной характеристикой. Во многих проектах этот кластер состоит из сотен или даже тысяч серверов, поэтому «тюнинг» процессорной нагрузки на кластере приложений оказывается более чем оправданным экономически (миллион долларов в нашем случае).



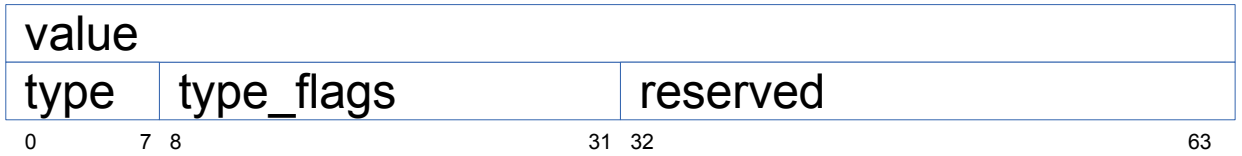




Что дальше?

- PHP 7.0
  - Оптимизация структур данных
- PHP 7.1
  - Анализатор потоков данных
  - Вывод типов
  - Глобальный оптимизатор для байт-кода PHP
  - Оптимизация и Специализация интерпретатора

zval



- IS_TYPE_CONSTANT
- IS_TYPE_REFCOUNTED
- IS_TYPE_COLLECTABLE
- IS_TYPE_COPYABLE
- IS_TYPE_IMMUTABLE

scalars

refcounted

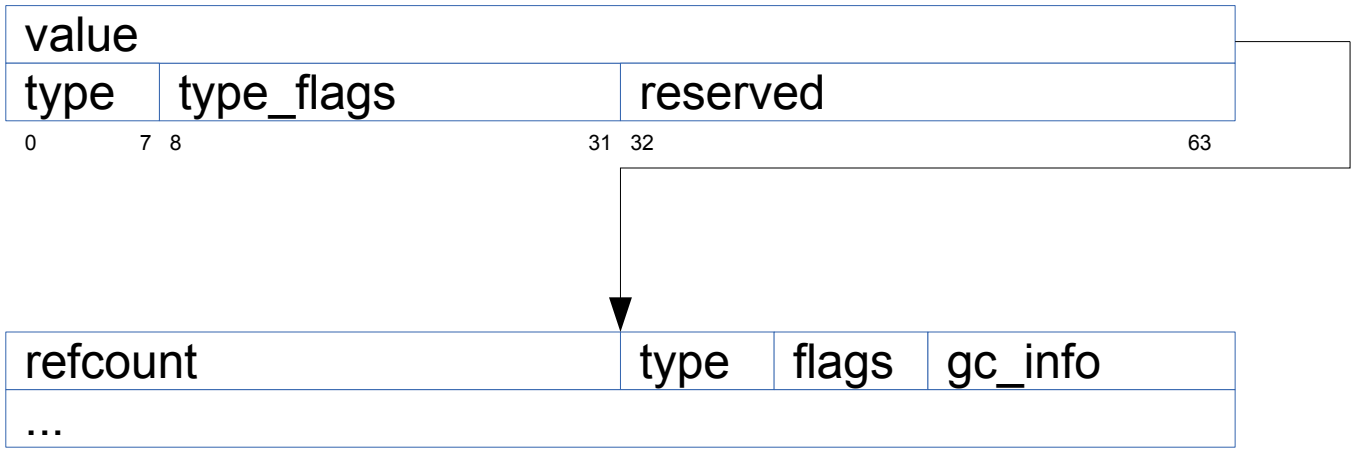
- IS_UNDEF
- IS_NULL
- IS_FALSE
- IS_TRUE
- IS_LONG
- IS_DOUBLE
- IS_STRING
- IS_ARRAY
- IS_OBJECT
- IS_RESOURCE
- IS_REFERENCE
- IS_INDIRECT
- IS_PTR

new type

old IS_BOOL

new type

zval (refcounted)



- IS_STRING
- IS_ARRAY
- IS_OBJECT
- IS_RESOURCE
- IS_REFERENCE

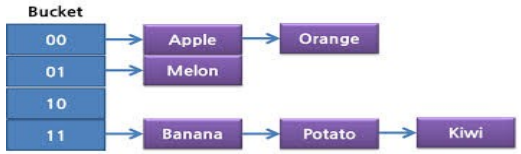
zval (string)

- IS_STR_PERSISTENT
- IS_STR_INTERNED
- IS_STR_PERMANENT

zval (array)

- IS_ARRAY_IMMUTABLE

zval (array) / HashTable



value		
type	flags	reserved

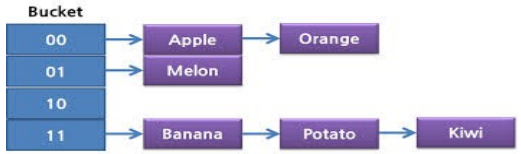
0 7 8 31 32 63

refcount	type	flags	gc_info
flags	nTableMask		
arData			
nNumUsed	nNumOfElem		
nTableSize	nInternalPtr		
nNextFreeElement			
pDestructor			

Bucket index N
...
Bucket index 0
Bucket 0
hash_val
key
val
...
Bucket N
hash_val
key
val

string, rc=1
«hello»

HashTable (PHP 5.*)



HashTable	
nTableSize	nTableMask
nNumOfElem	
nNextFreeElement	
pInternalPointer	
pListHead	
pListTail	
arBuckets	
pDestructor	

Bucket
hash_val
nKeyLenght
pData
pDataPtr
pListNext
pListPrev
pNext
pPrev
arKey

Bucket
hash_val
nKeyLenght
pData
pDataPtr
pListNext
pListPrev
pNext
pPrev
arKey

Bucket*
Bucket*
...
Bucket*

«hello»

zval
value
type

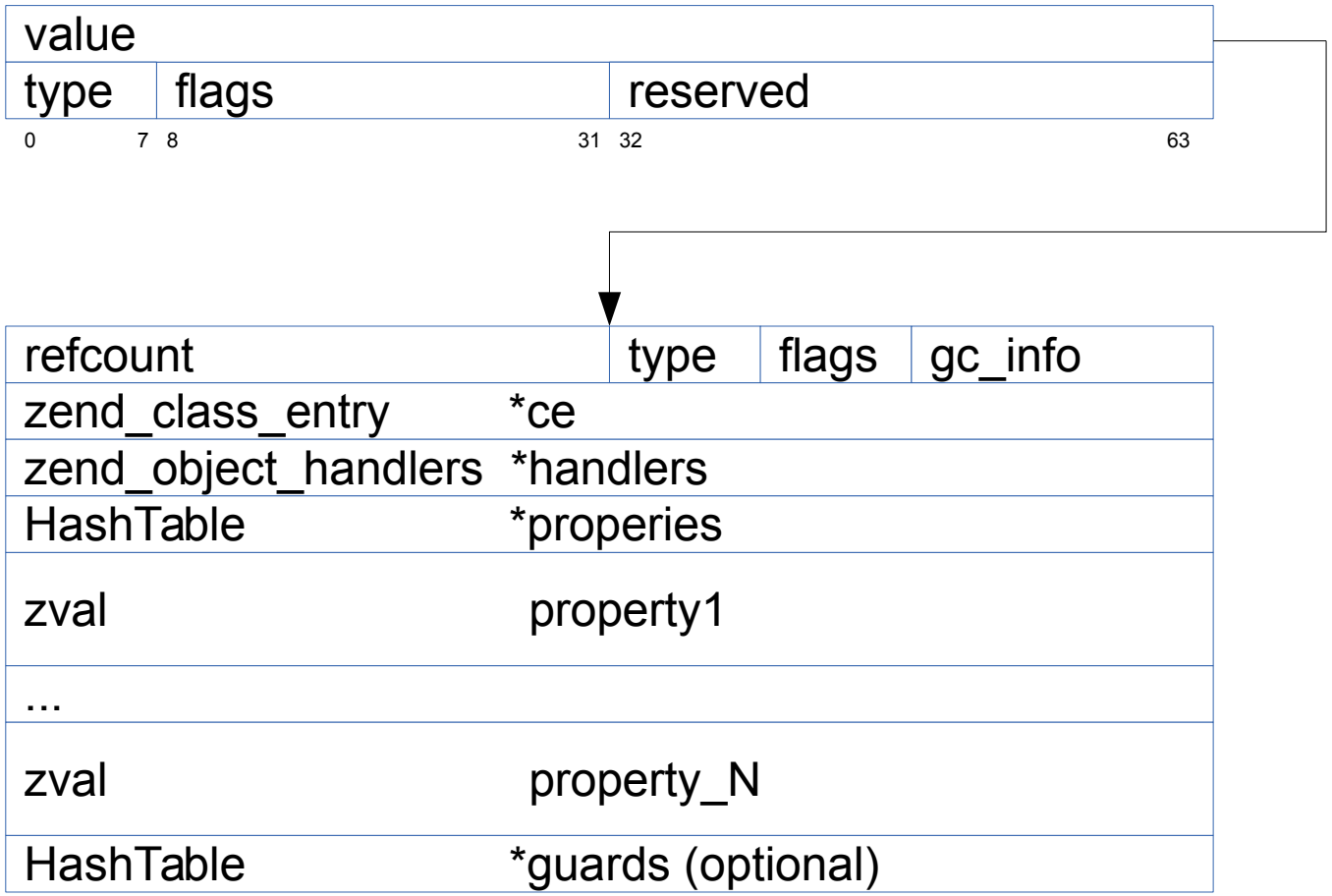
Immutable Arrays (Неизменяемые массивы)

```
$a = array();
for ($i = 0; $i < 1000000; $i++) $a[$i] = array("hello");
echo memory_get_usage(true);
```

	PHP 5.6	PHP 7
Memory Usage	428 MB	34 MB
Time	0.49 sec	0.06 sec

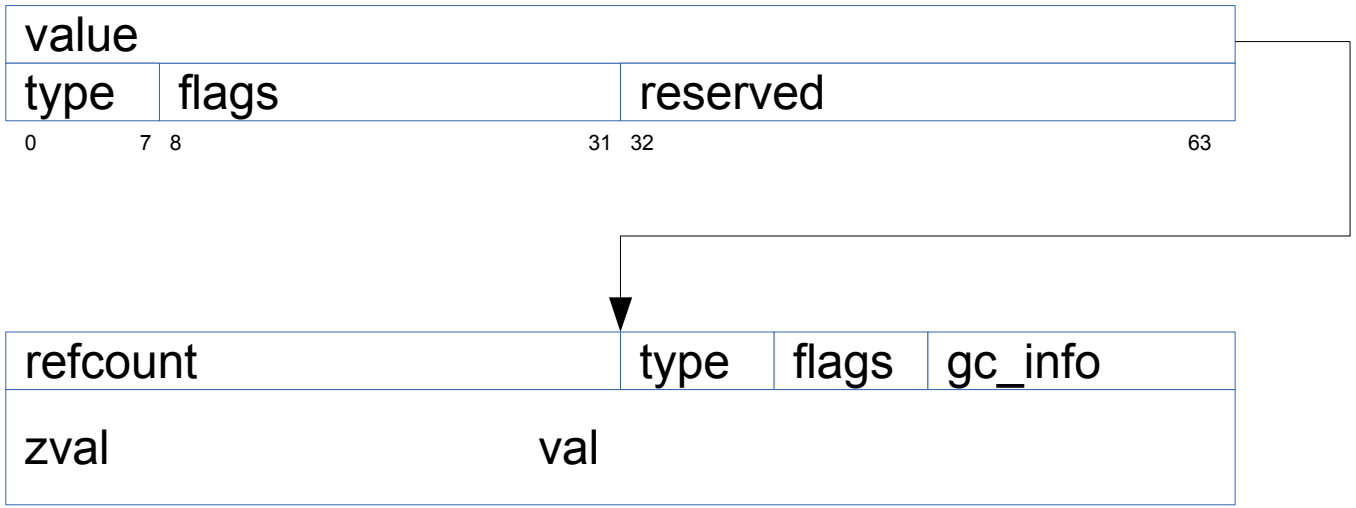
```
if (in array($color, array(“red”, “yellow”, “green”)) {
    ...
}
```

zval (object/PHP 7)

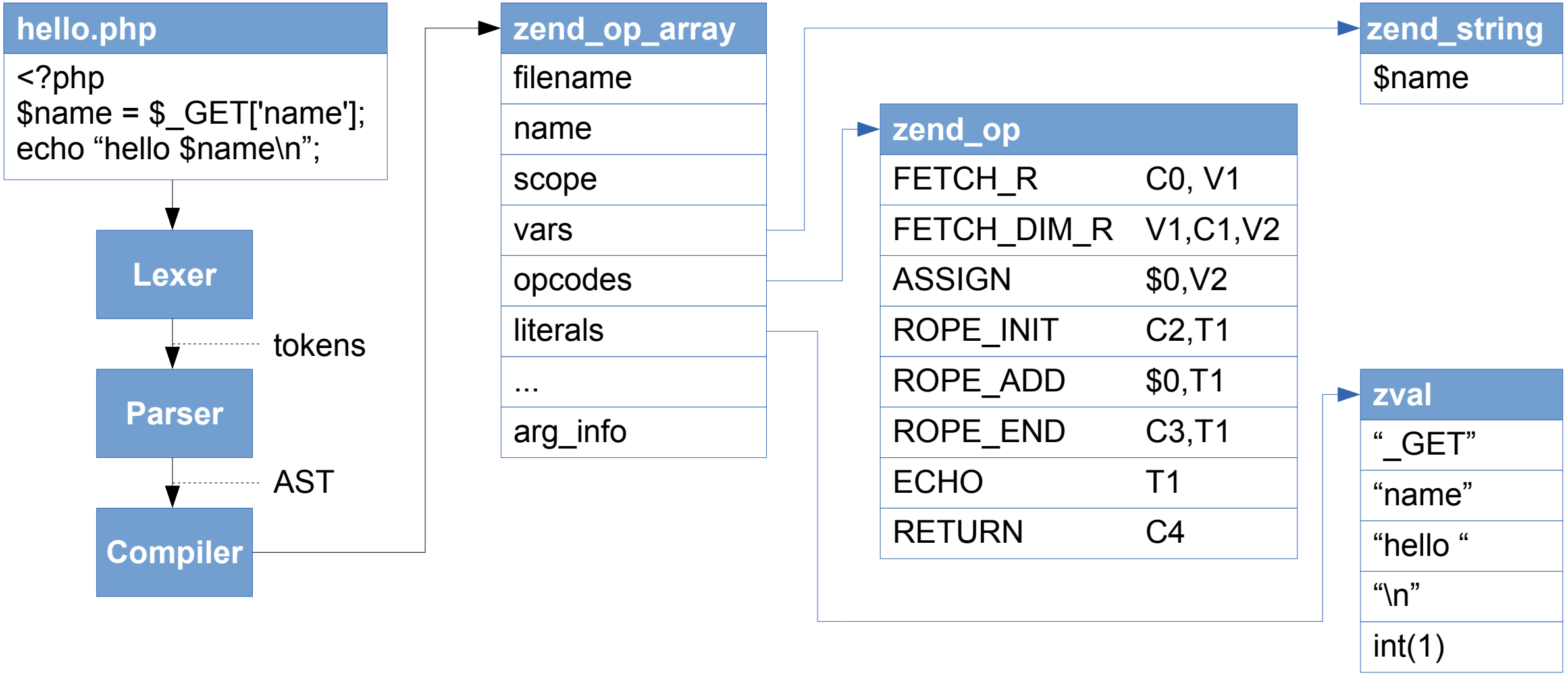


- IS_OBJ_DTOR_CALLED
- IS_OBJ_FREE_CALLED
- IS_OBJ_USE_GUARDS
- IS_OBJ_HAS_GUARDS

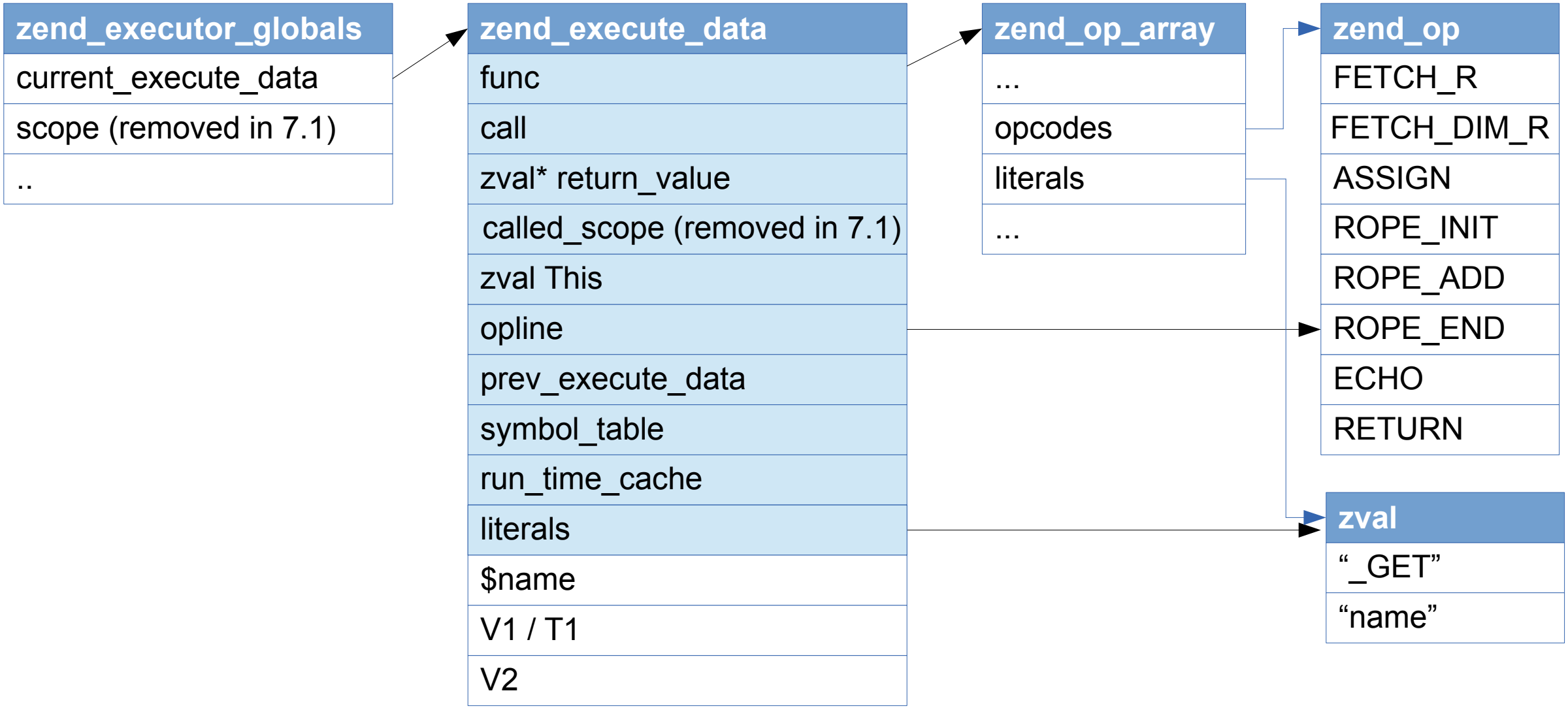
zval (reference)



Компиляция PHP 7



Структуры Данных PHP 7 Времени Исполнения



Интерпретация PHP 7

```
register zend_execute_data *exexute_data
        __asm__(«%r14»);
register zend_op *opline
        __asm__(«%r15»);

void execute_ex(zend_execute_data *ex)
{
    exexute_data = ex;
    opline = exexcute_data->opline;
    do {
        opline->handler();
    } while (opline);
}
```

```
ZEND_VM_HANDLER(1, ZEND_ADD,
                CONST|TMPVAR|CV, CONST|TMPVAR|CV)
{
        zval *op1, *op2, *result;

        op1 = GET_OP1_ZVAL_PTR(BP_VAR_R);
        op2 = GET_OP2_ZVAL_PTR(BP_VAR_R);
        if (Z_TYPE_P(op1)== IS_LONG) {
            if (Z_TYPE_P(op2) == IS_LONG) {
                result = EX_VAR(opline->result.var);
                fast_long_add_function(result, op1, op2);
            } else if (Z_TYPE_P(op2) == IS_DOUBLE) {
                ...
        } else{
            …
        }
        opline++;
}
```

Интерпретация PHP 7

```
register zend_execute_data *exexute_data
        __asm__(«%r14»);
register zend_op *opline
        __asm__(«%r15»);

void execute_ex(zend_execute_data *ex)
{
    exexute_data = ex;
    opline = exexcute_data->opline;
    do {
        opline->handler();
    } while (opline);
}
```

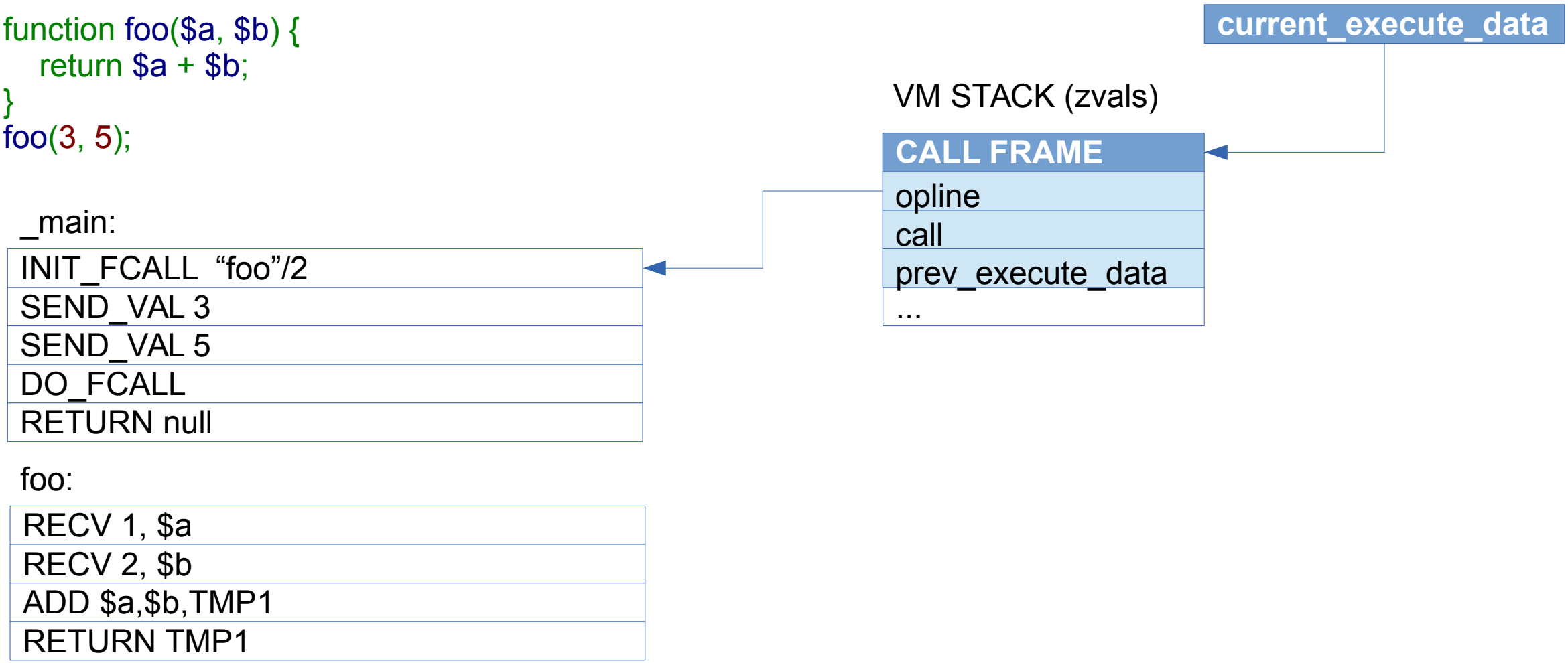
```
static void ZEND_ADD_SPEC_TMPVAR_CONST(void)

{
        zval *op1, *op2, *result;

        op1 = ZEND_CALL_VAR(execute_data, opline->op1.var);
        op2 = execute_data->literals[opline->op2.num];
        if (Z_TYPE_P(op1)== IS_LONG) {
            if (Z_TYPE_P(op2) == IS_LONG) {
                result = EX_VAR(opline->result.var);
                fast_long_add_function(result, op1, op2);
            } else if (Z_TYPE_P(op2) == IS_DOUBLE) {
                ...
        } else{
            …
        }
        opline++;
}
```

VM Calling Convention (PHP 7)

```php
function foo($a, $b) {
    return $a + $b;
}
foo(3, 5);
```



VM Calling Convention (PHP 7)

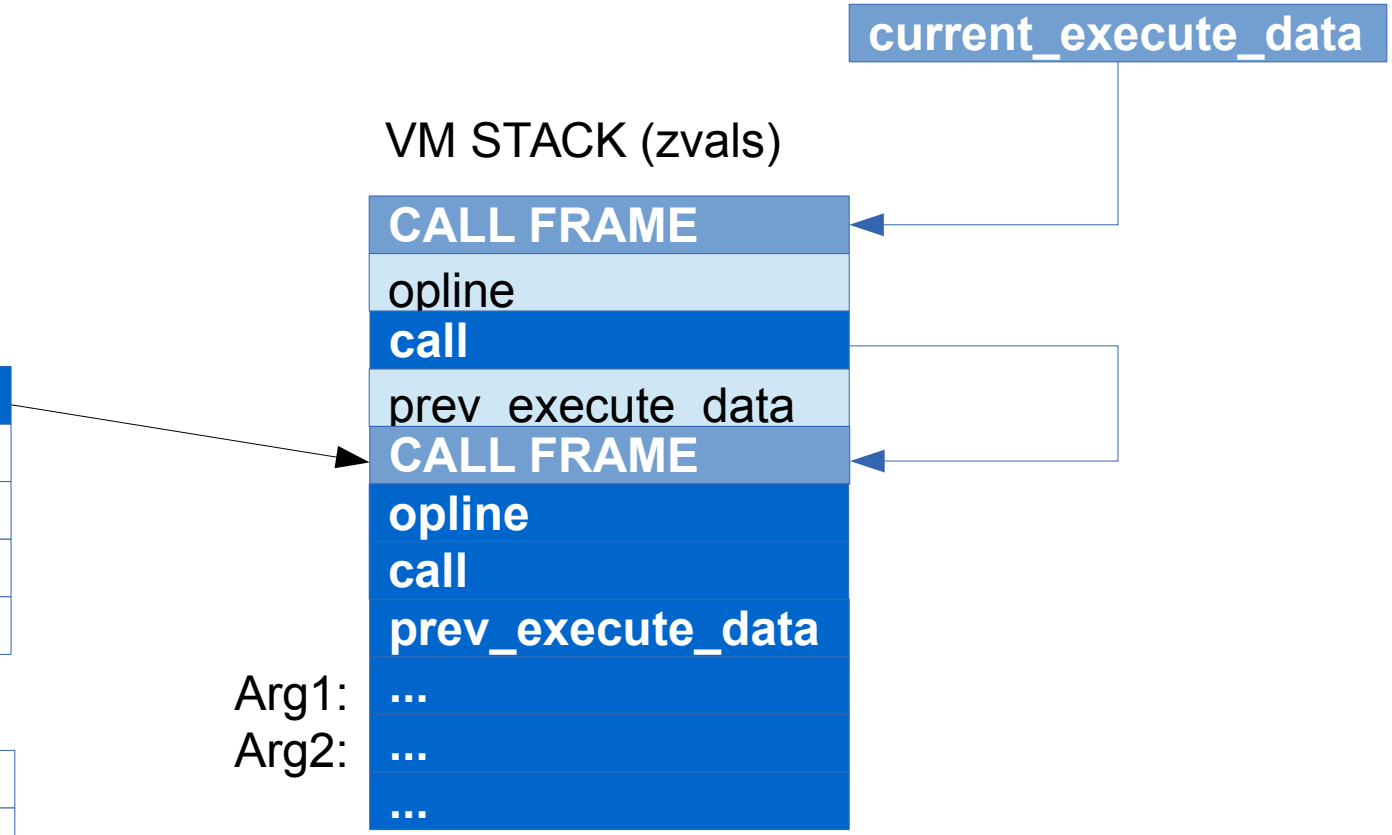
```php
function foo($a, $b) {
    return $a + $b;
}
foo(3, 5);
```

_main:

INIT_FCALL “foo”/2
SEND_VAL 3
SEND_VAL 5
DO_FCALL
RETURN null

foo:

RECV 1, $a
RECV 2, $b
ADD $a,$b,TMP1
RETURN TMP1



VM Calling Convention (PHP 7)

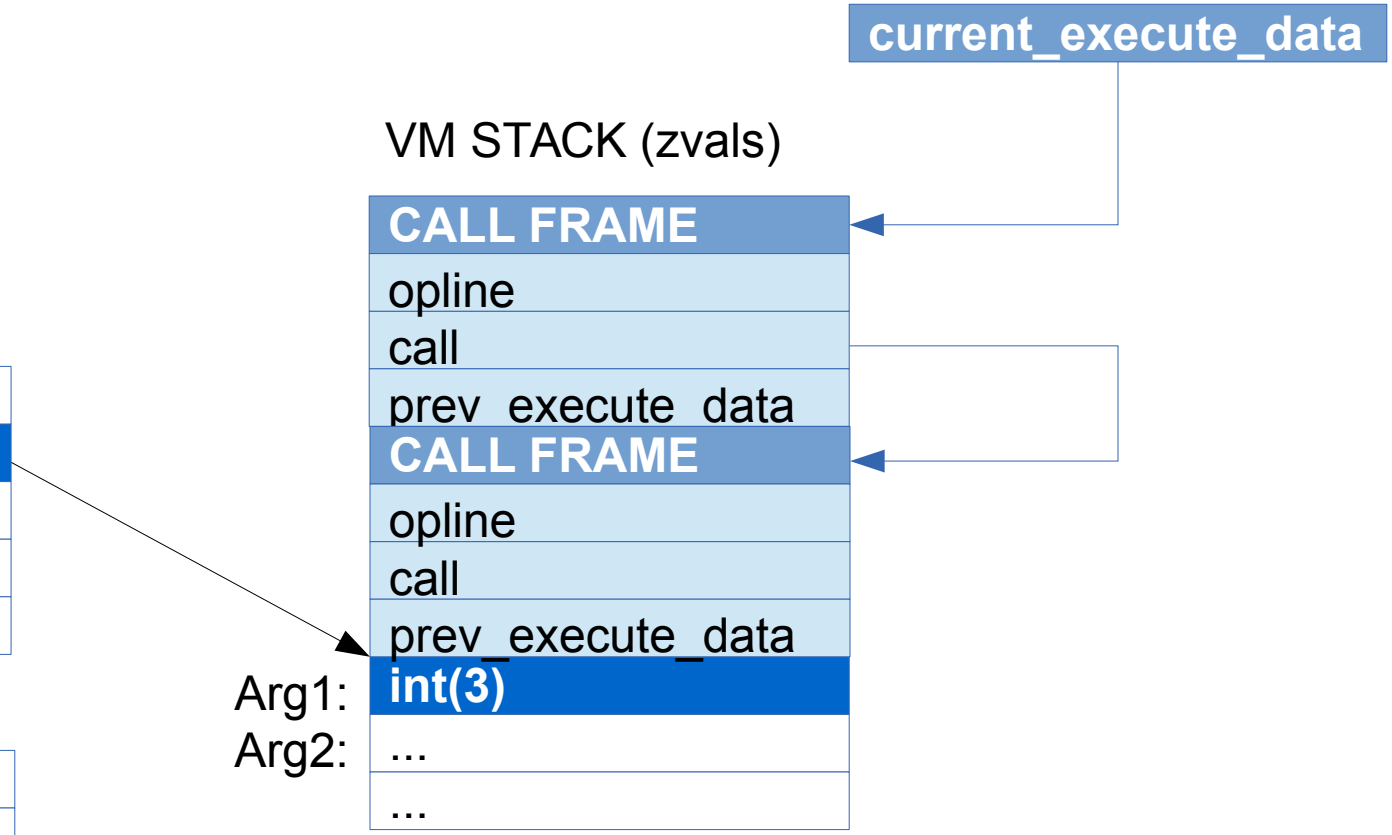
```php
function foo($a, $b) {
    return $a + $b;
}
foo(3, 5);
```

_main:

INIT_FCALL “foo”/2
SEND_VAL 3
SEND_VAL 5
DO_FCALL
RETURN null

foo:

RECV 1, $a
RECV 2, $b
ADD $a,$b,TMP1
RETURN TMP1



VM Calling Convention (PHP 7)

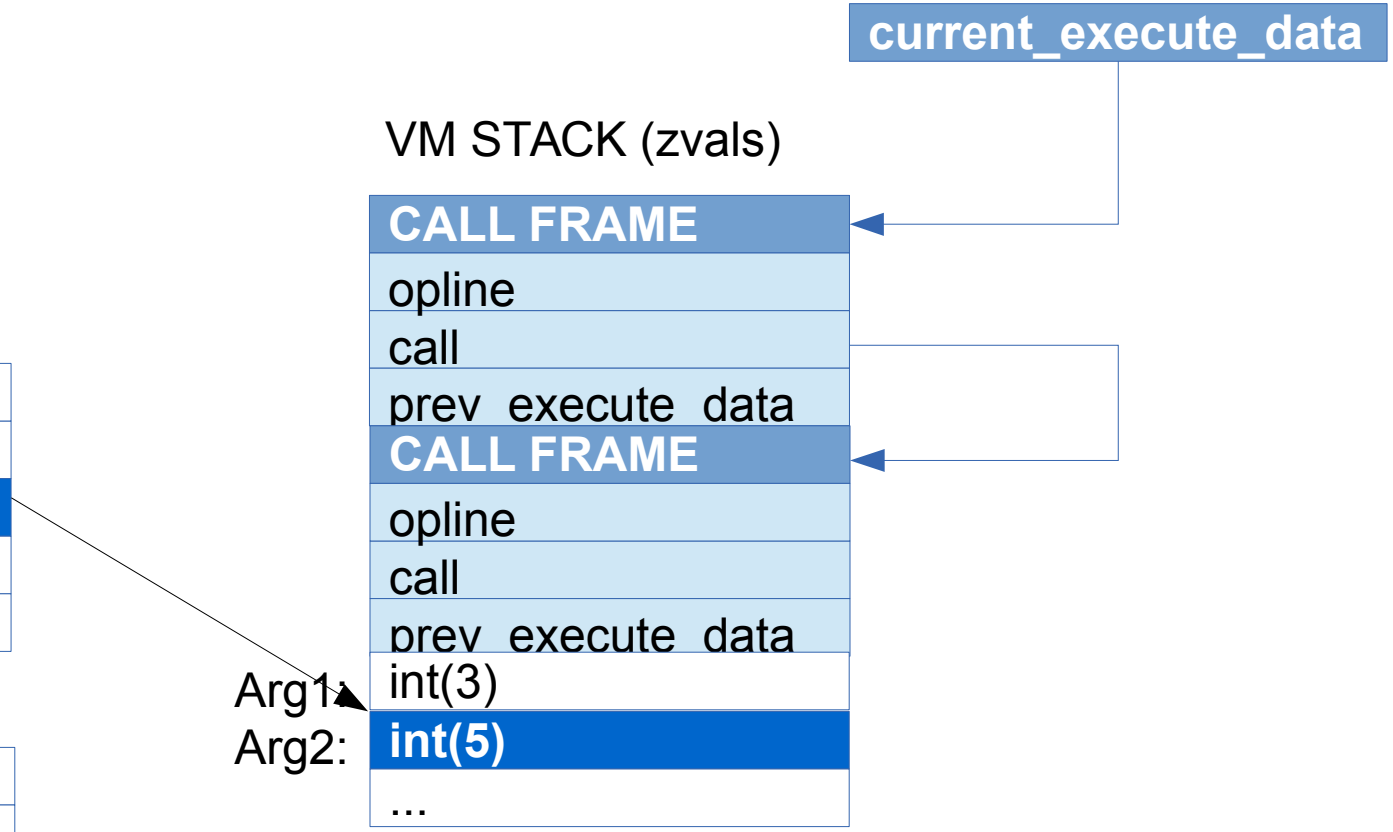
```php
function foo($a, $b) {
    return $a + $b;
}
foo(3, 5);
```

_main:

INIT_FCALL “foo”/2
SEND_VAL 3
SEND_VAL 5
DO_FCALL
RETURN null

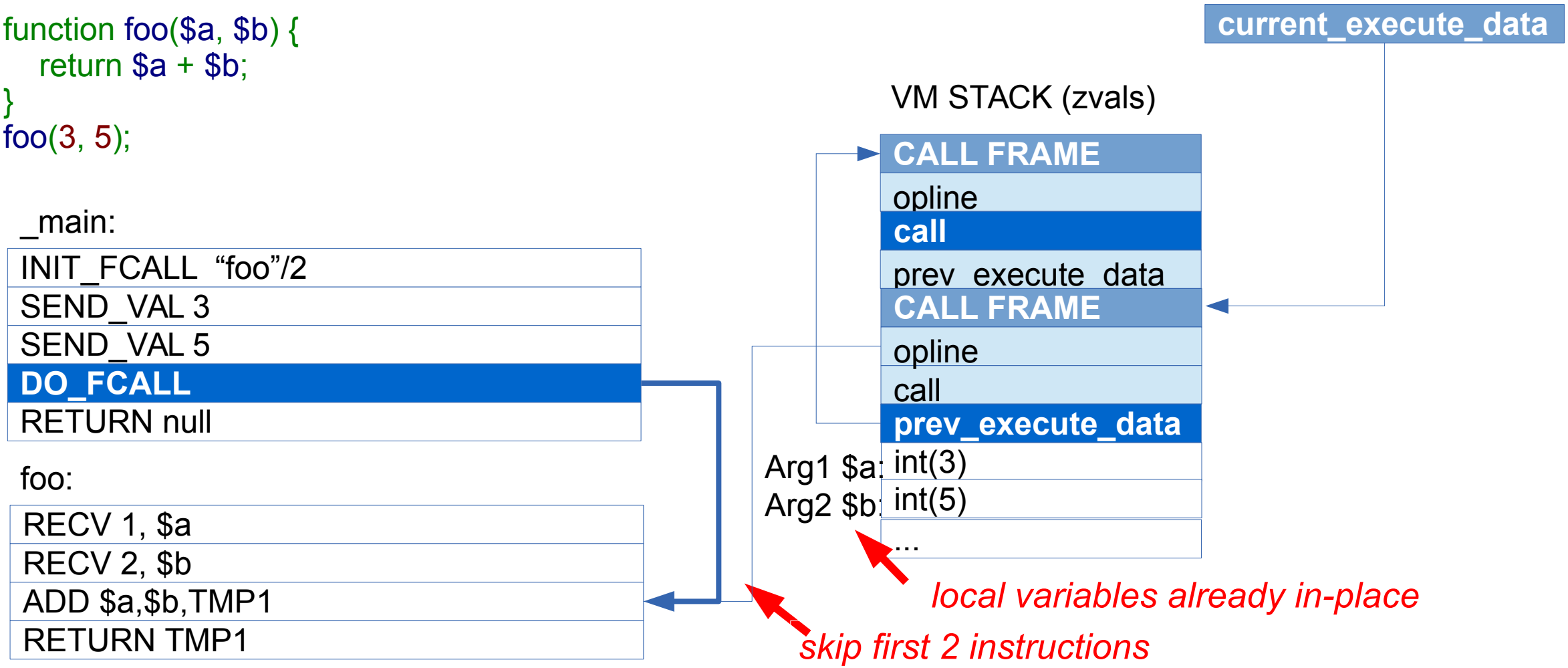
foo:

RECV 1, $a
RECV 2, $b
ADD $a,$b,TMP1
RETURN TMP1



VM Calling Convention (PHP 7)

```
function foo($a, $b) {
    return $a + $b;
}
foo(3, 5);
```



VM Calling Convention (PHP 7)

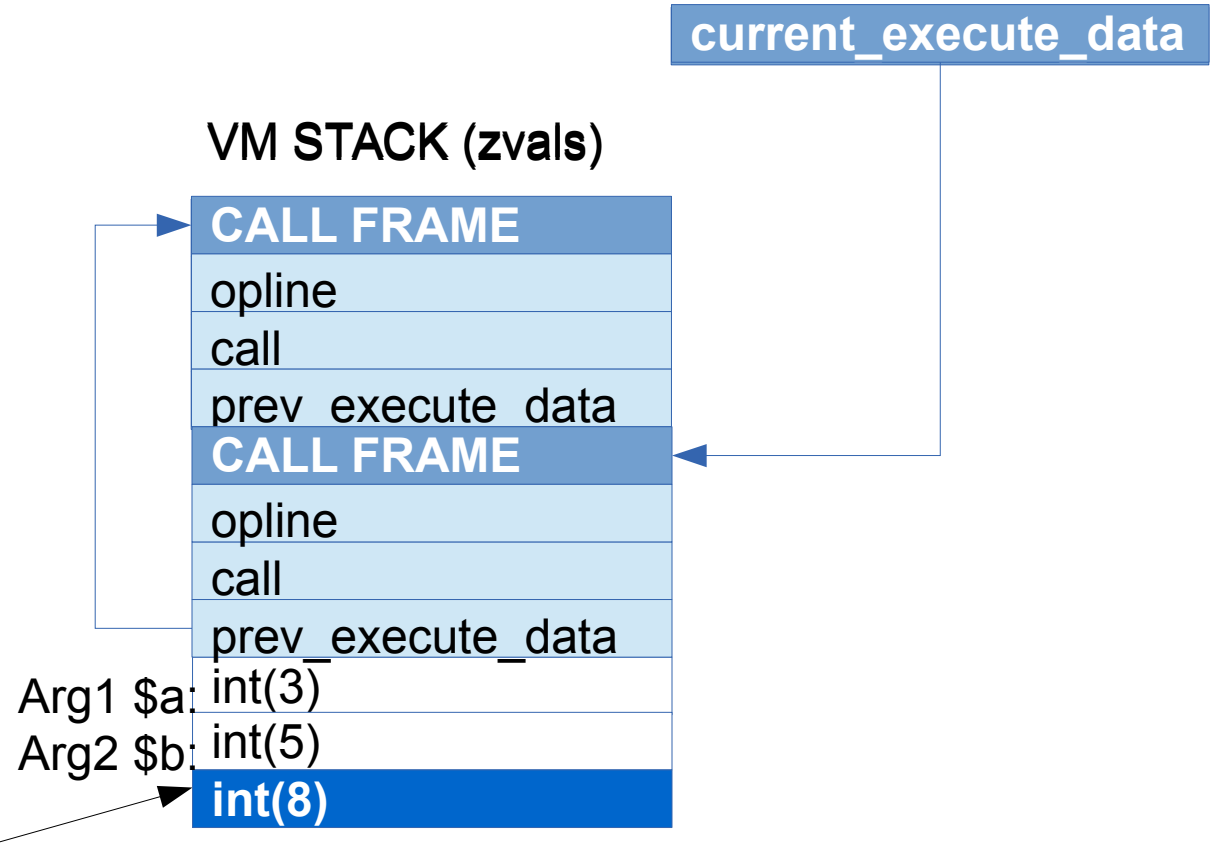
```
function foo($a, $b) {
    return $a + $b;
}
foo(3, 5);
```

_main:

INIT_FCALL “foo”/2
SEND_VAL 3
SEND_VAL 5
DO_FCALL
RETURN null

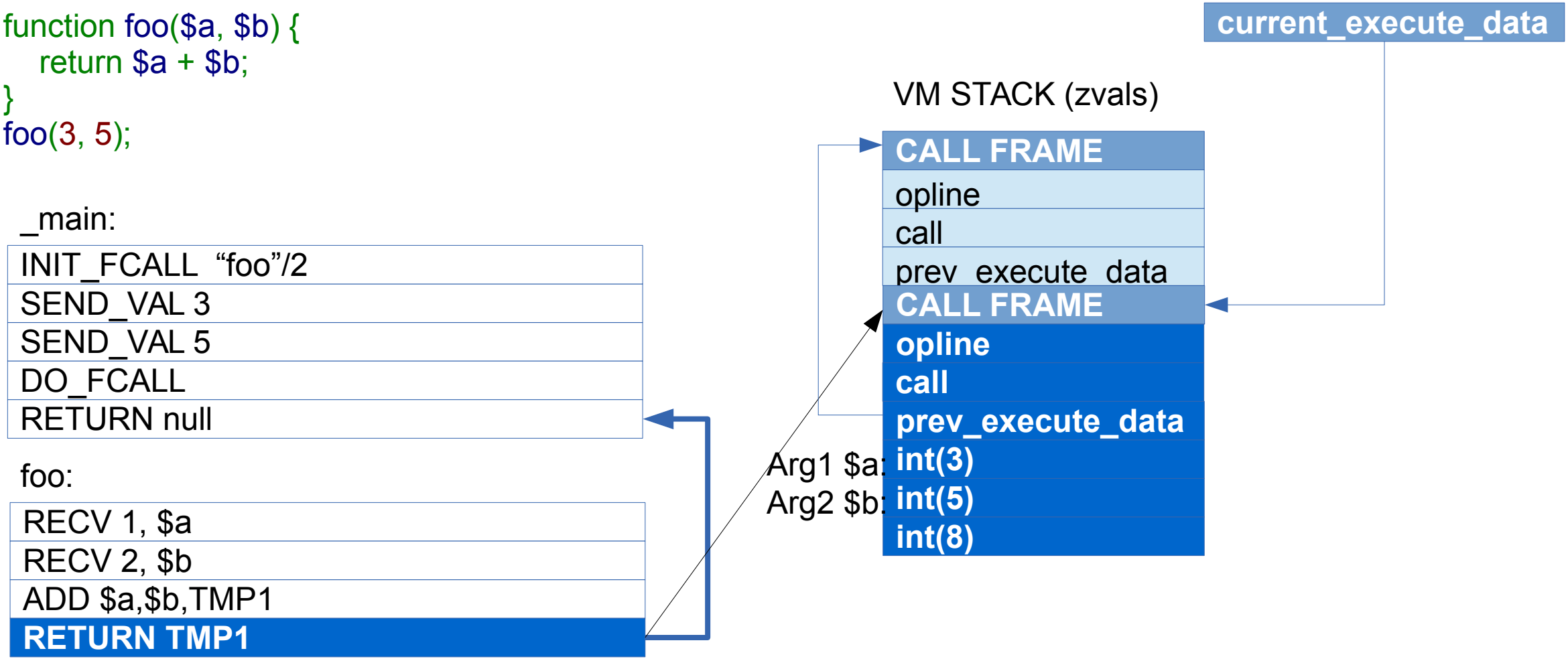
foo:

RECV 1, $a
RECV 2, $b
ADD $a,$b,TMP1
RETURN TMP1



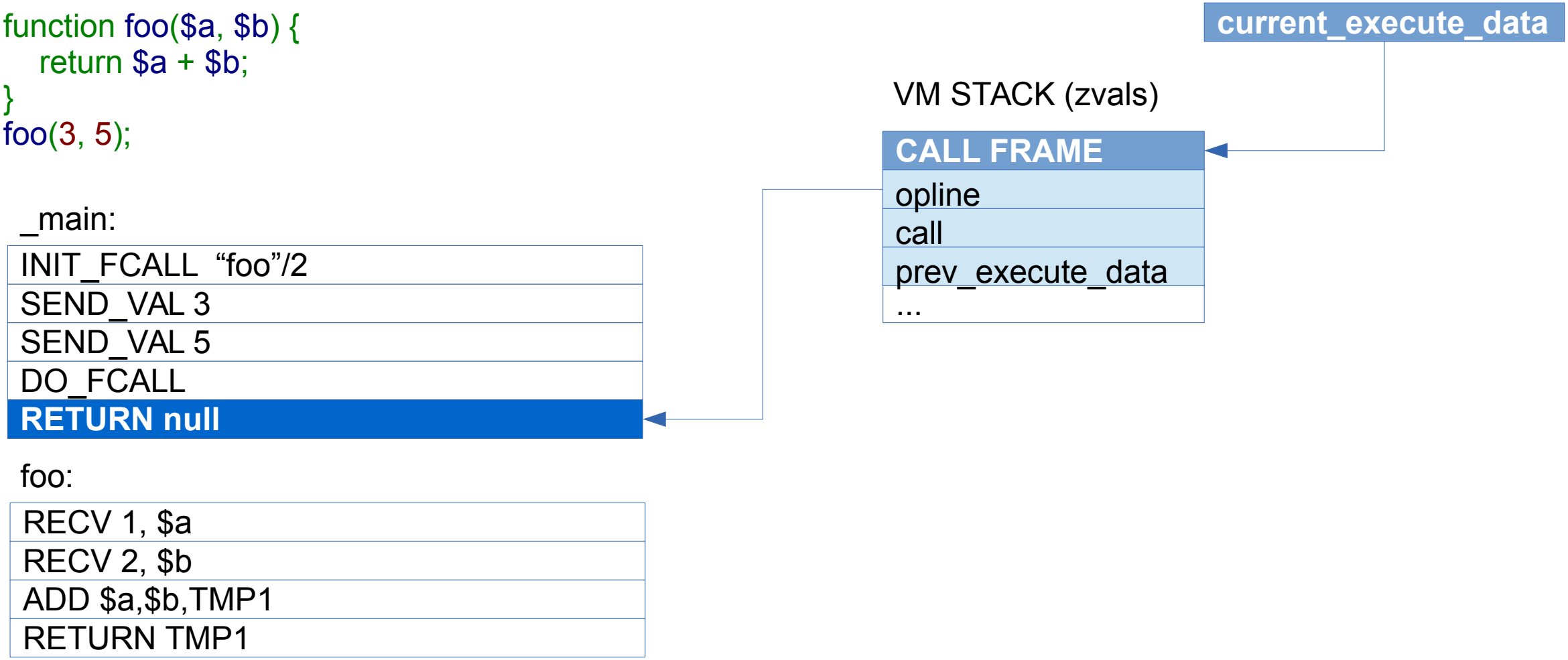
VM Calling Convention (PHP 7)

```
function foo($a, $b) {
    return $a + $b;
}
foo(3, 5);
```



VM Calling Convention (PHP 7)

```
function foo($a, $b) {
    return $a + $b;
}
foo(3, 5);
```



Что дальше?

- PHP 7.0
  - Оптимизация структур данных
- PHP 7.1
  - Анализатор потоков данных
  - Вывод типов
  - Глобальный оптимизатор для байт-кода PHP
  - Оптимизация и Специализация интерпретатора

PHP 7.1

- Первый Alpha релиз в июне 2016
- Feature Freeze в июле
- GA релиз запланирован на ноябрь 2016

- Nullable types - function foo(?Node $x): ?Node;
- Void return type - function foo(): void;
- Keys in list() - foreach ($points as list(«x»=>$x, «y»=>$y))
- Class constants visibility - private const X = 42;
- Negative string offsets - $a = “abcd”; var_dump($a[-2]);
- Invalid numeric strings - 5 * “orange”
- Closure::fromCallable()

PHP 7.1 Optimizer (script)

```php
<?php
function sum() {
        $sum = 0;
        for ($i = 0; $i < 100; $i++) {
                $sum = $sum + $i;
        }
        return $sum;
}
```

PHP 7.1 Optimizer (bytecode)

```php
<?php
function sum() {
        $sum = 0;
        for ($i = 0; $i < 100; $i++) {
                $sum = $sum + $i;
        }
        return $sum;
}
```

```
  ASSIGN          $sum, 0
  ASSIGN          $i, 0
  JMP L1
L0:
  ADD             $sum, $i -> T2
  ASSIGN          $sum, T2
  POST_INC        $i -> T4
  FREE            T4
L1:
  IS_SMALLER      $i, 100 -> T5
  JMPNZ           T5, L0
  RETRUN          $sum
  RETUTN          null
```

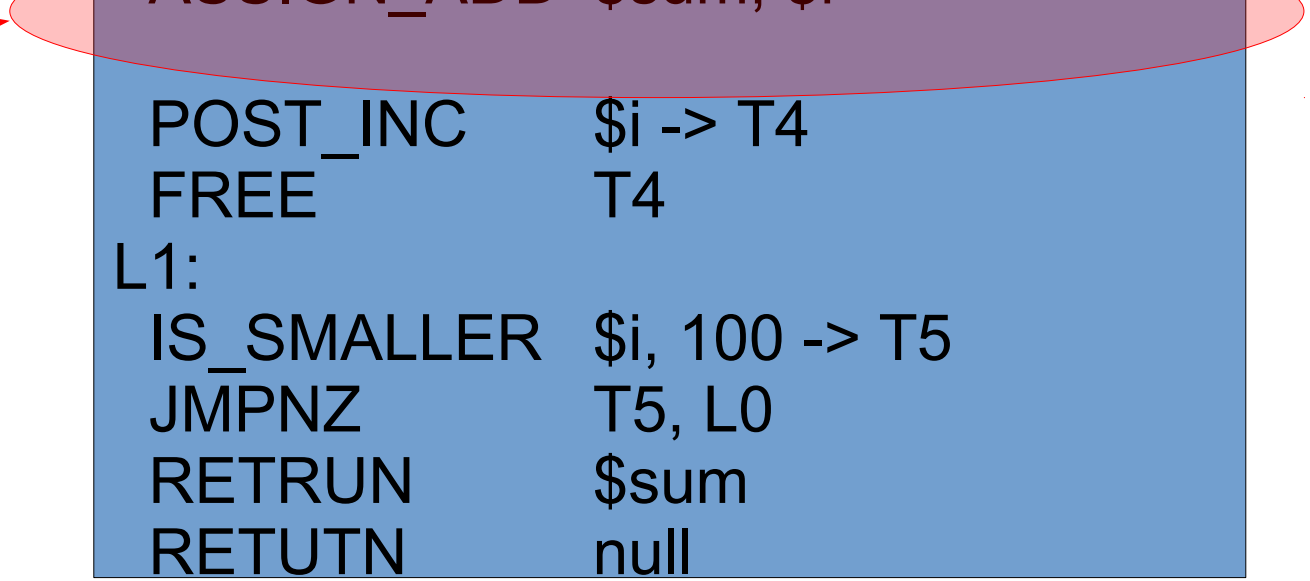
PHP 7.1 Optimizer (trivial optimization)

```php
<?php
function sum() {
        $sum = 0;
        for ($i = 0; $i < 100; $i++) {
                $sum = $sum + $i;
        }
        return $sum;
}
```

```
  ASSIGN      $sum, 0
  ASSIGN      $i, 0
  JMP L1
L0:
  ADD         $sum, $i -> T2
  ASSIGN      $sum, T2
  POST_INC    $i -> T4
  FREE        T4
L1:
  IS_SMALLER  $i, 100 -> T5
  JMPNZ       T5, L0
  RETRUN      $sum
  RETUTN      null
```

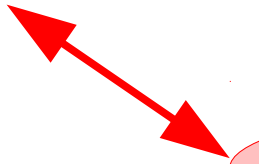
PHP 7.1 Optimizer (trivial optimization)

```
<?php
function sum() {
        $sum = 0;
        for ($i = 0; $i < 100; $i++) {
                $sum += $i;
        }
        return $sum;
}
```



PHP 7.1 Optimizer (trivial optimization)

```php
<?php
function sum() {
        $sum = 0;
        for ($i = 0; $i < 100; $i++) {
                $sum = $sum + $i;
        }
        return $sum;
}
```

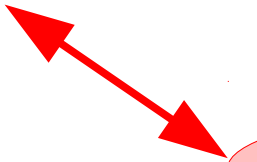


```
  ASSIGN          $sum, 0
  ASSIGN          $i, 0
  JMP L1
L0:
  ASSIGN_ADD      $sum, $i

  POST_INC        $i -> T4
  FREE            T4
L1:
  IS_SMALLER      $i, 100 -> T5
  JMPNZ           T5, L0
  RETRUN          $sum
  RETUTN          null
```

PHP 7.1 Optimizer (trivial optimization)

```php
<?php
function sum() {
        $sum = 0;
        for ($i = 0; $i < 100; ++$i) {
                $sum = $sum + $i;
        }
        return $sum;
}
```



```
  ASSIGN          $sum, 0
  ASSIGN          $i, 0
  JMP L1
L0:
  ASSIGN_ADD      $sum, $i

  PRE_INC         $i

L1:
  IS_SMALLER      $i, 100 -> T5
  JMPNZ           T5, L0
  RETRUN          $sum
  RETUTN          null
```

PHP 7.1 Optimizer (trivial optimization)

```php
<?php
function sum() {
        $sum = 0;
        for ($i = 0; $i < 100; $i++) {
                $sum = $sum + $i;
        }
        return $sum;
}
```

```
  ASSIGN          $sum, 0
  ASSIGN          $i, 0
  JMP L1
L0:
  ASSIGN_ADD      $sum, $i

  PRE_INC         $i

L1:
  IS_SMALLER      $i, 100 -> T5
  JMPNZ           T5, L0
  RETRUN          $sum
  RETUTN          null
```

PHP 7.1 Optimizer (trivial optimization)

```php
<?php
function sum() {
        $sum = 0;
        for ($i = 0; $i < 100; $i++) {
                $sum = $sum + $i;
        }
        return $sum;
}
```

```
  ASSIGN          $sum, 0
  ASSIGN          $i, 0
  JMP L1
L0:
  ASSIGN_ADD      $sum, $i

  PRE_INC         $i

L1:
  IS_SMALLER      $i, 100 -> T5
  JMPNZ           T5, L0
  RETRUN          $sum
```

PHP 7.1 Optimizer (Control Flow Graph)

```php
<?php
function sum() {
        $sum = 0;
        for ($i = 0; $i < 100; $i++) {
                $sum = $sum + $i;
        }
        return $sum;
}
```

```
  ASSIGN          $sum, 0
  ASSIGN          $i, 0
  JMP L1
L0:
  ASSIGN_ADD      $sum, $i

  PRE_INC         $i

L1:
  IS_SMALLER      $i, 100 -> T5
  JMPNZ           T5, L0
  RETRUN          $sum
```

PHP 7.1 Optimizer (Control Flow Graph)

```php
<?php
function sum() {
        $sum = 0;
        for ($i = 0; $i < 100; $i++) {
                $sum = $sum + $i;
        }
        return $sum;
}
```

```
  ASSIGN          $sum, 0
  ASSIGN          $i, 0
  JMP L1
```

```
L0:
  ASSIGN_ADD      $sum, $i
  PRE_INC         $i
```

```
L1:
  IS_SMALLER      $i, 100 -> T5
  JMPNZ           T5, L0
  RETRUN          $sum
```

PHP 7.1 Optimizer (Control Flow Graph)

```php
<?php
function sum() {
        $sum = 0;
        for ($i = 0; $i < 100; $i++) {
                $sum = $sum + $i;
        }
        return $sum;
}
```

```
  ASSIGN          $sum, 0
  ASSIGN          $i, 0
  JMP L1
```

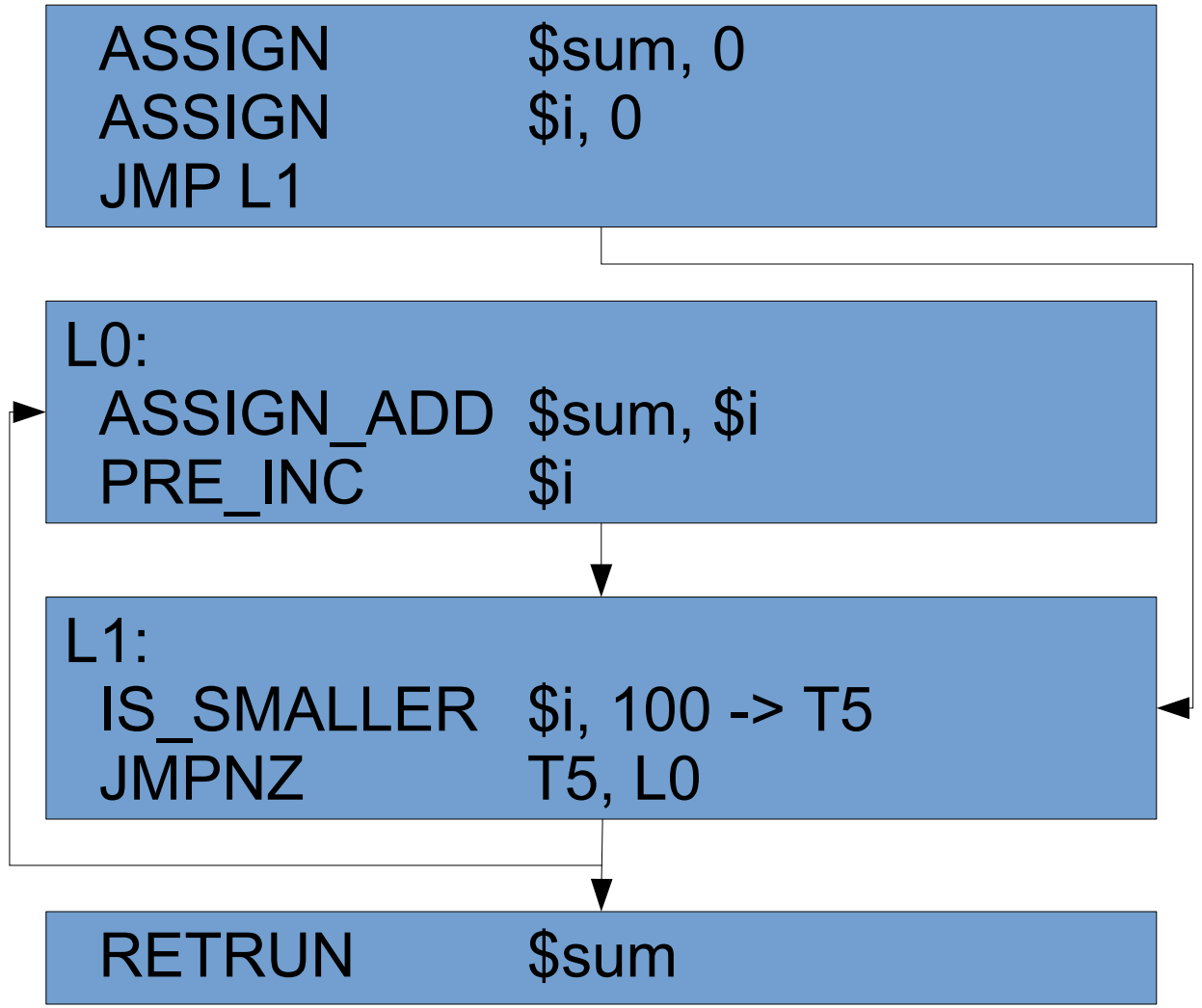
```
L0:
  ASSIGN_ADD      $sum, $i
  PRE_INC         $i
```

```
L1:
  IS_SMALLER      $i, 100 -> T5
  JMPNZ           T5, L0
```

```
  RETRUN          $sum
```

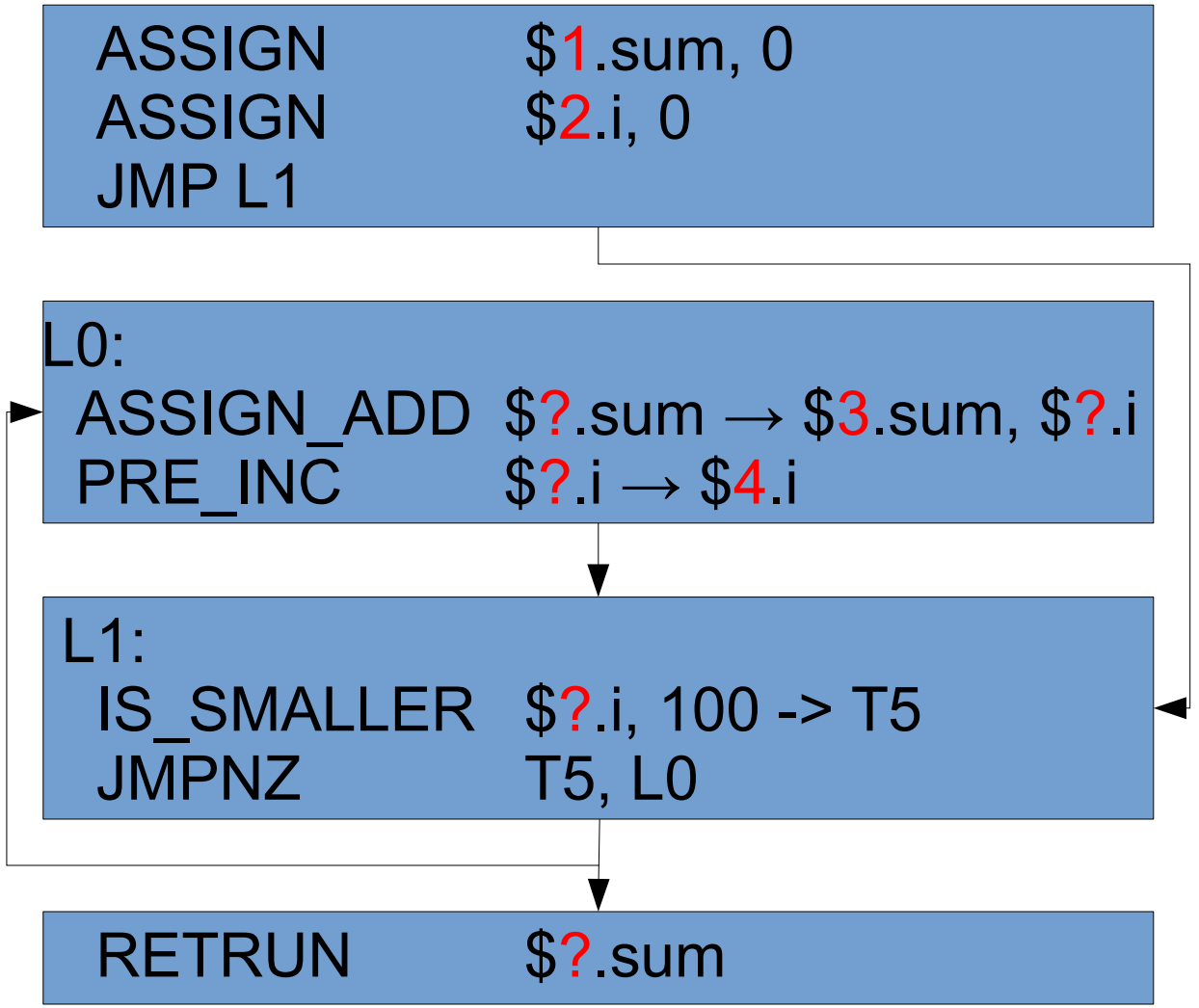
PHP 7.1 Optimizer (Control Flow Graph)

```php
<?php
function sum() {
        $sum = 0;
        for ($i = 0; $i < 100; $i++) {
                $sum = $sum + $i;
        }
        return $sum;
}
```



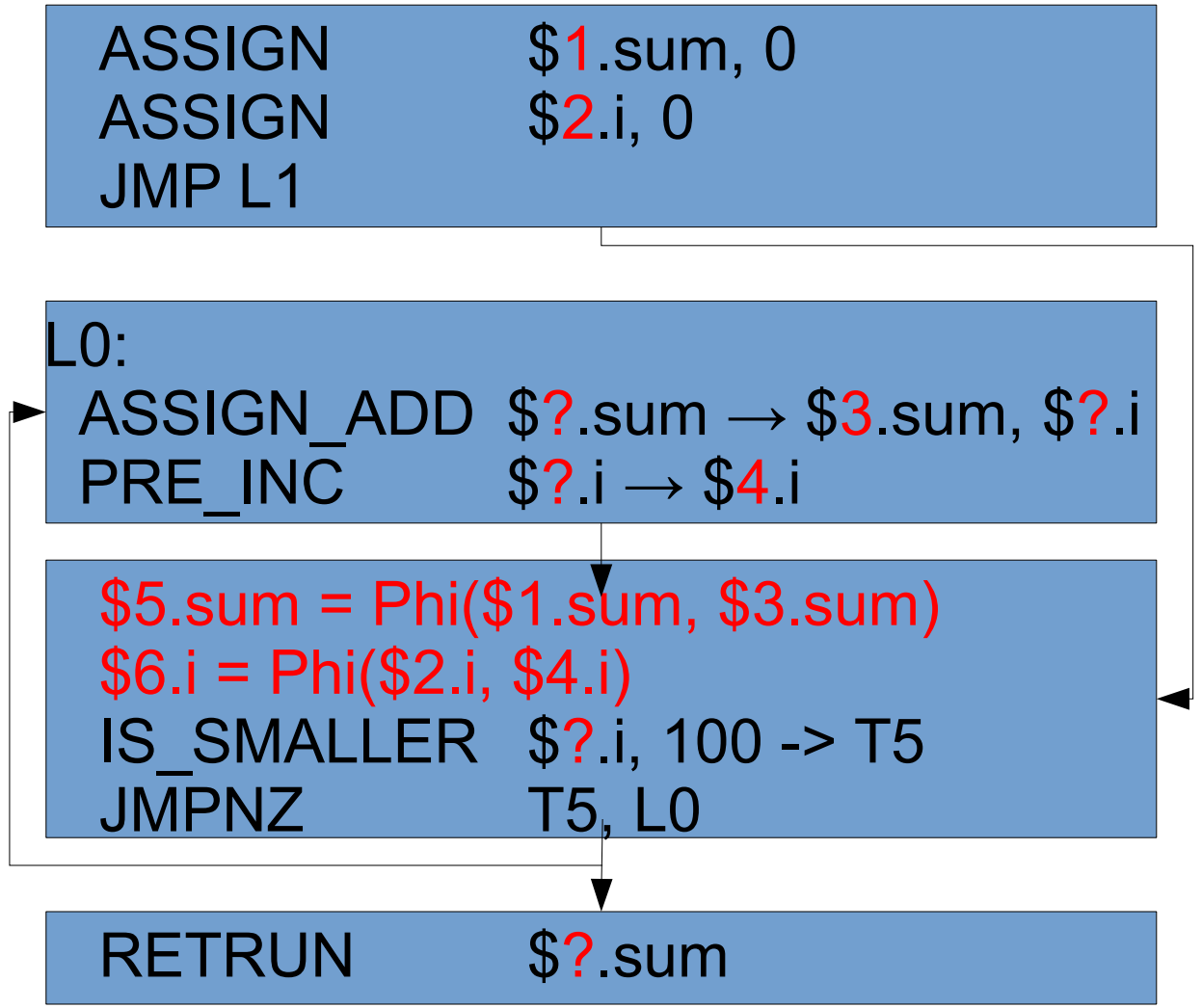
PHP 7.1 Optimizer (Static Single Assignmnt Form)

```php
<?php
function sum() {
        $sum = 0;
        for ($i = 0; $i < 100; $i++) {
                $sum = $sum + $i;
        }
        return $sum;
}
```



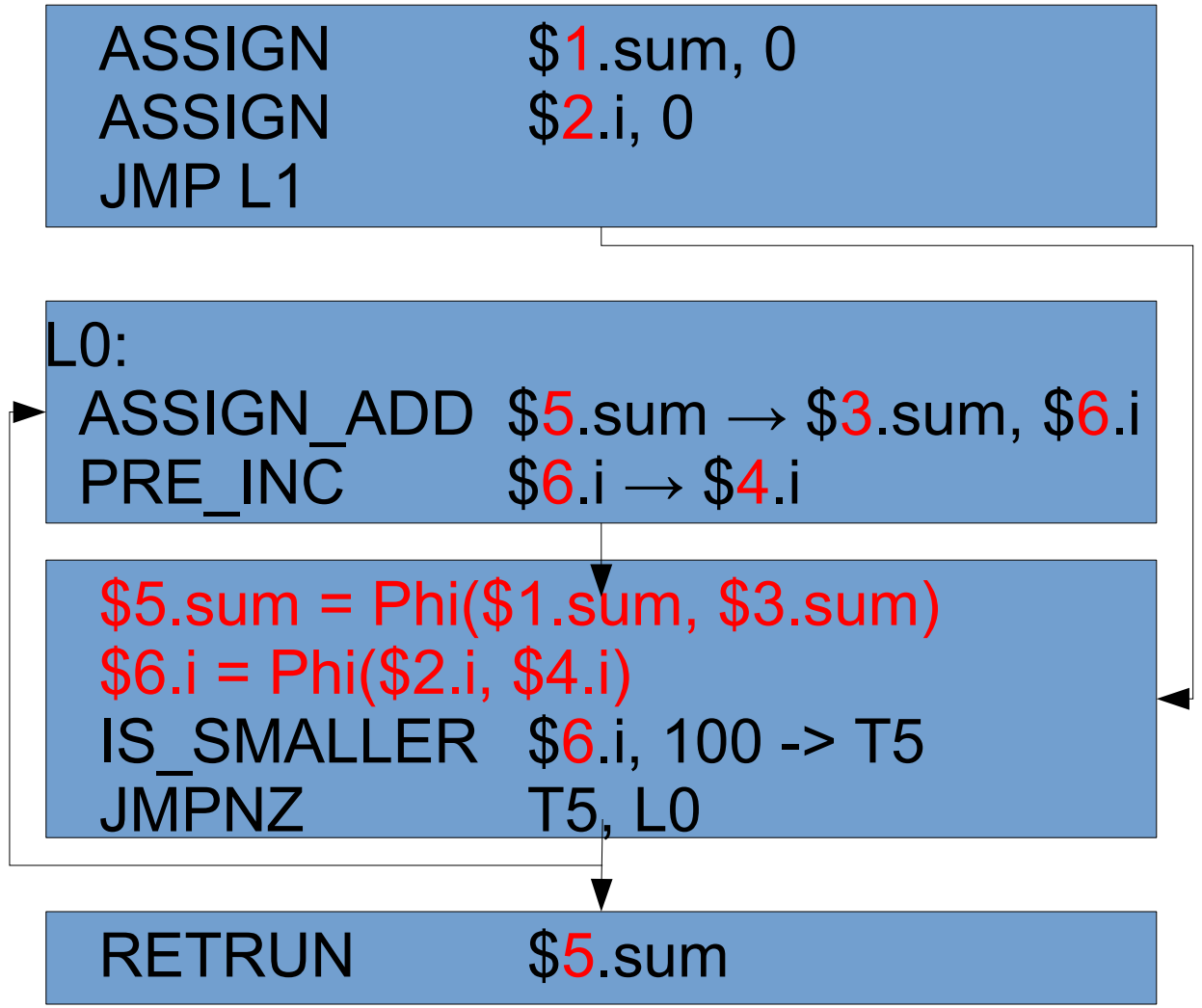
PHP 7.1 Optimizer (Static Single Assignmnt Form)

```php
<?php
function sum() {
        $sum = 0;
        for ($i = 0; $i < 100; $i++) {
                $sum = $sum + $i;
        }
        return $sum;
}
```



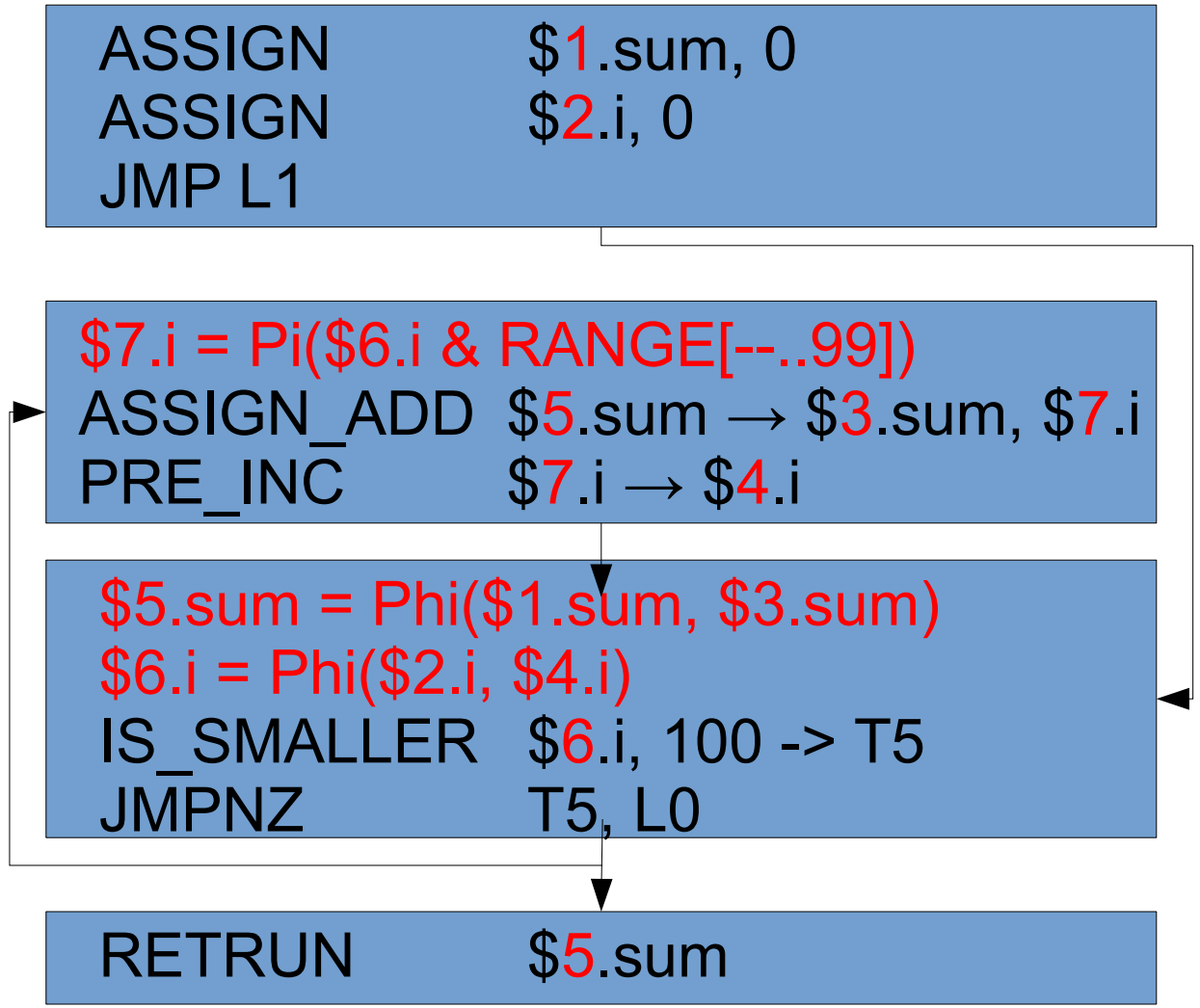
PHP 7.1 Optimizer (Static Single Assignmnt Form)

```php
<?php
function sum() {
        $sum = 0;
        for ($i = 0; $i < 100; $i++) {
                $sum = $sum + $i;
        }
        return $sum;
}
```

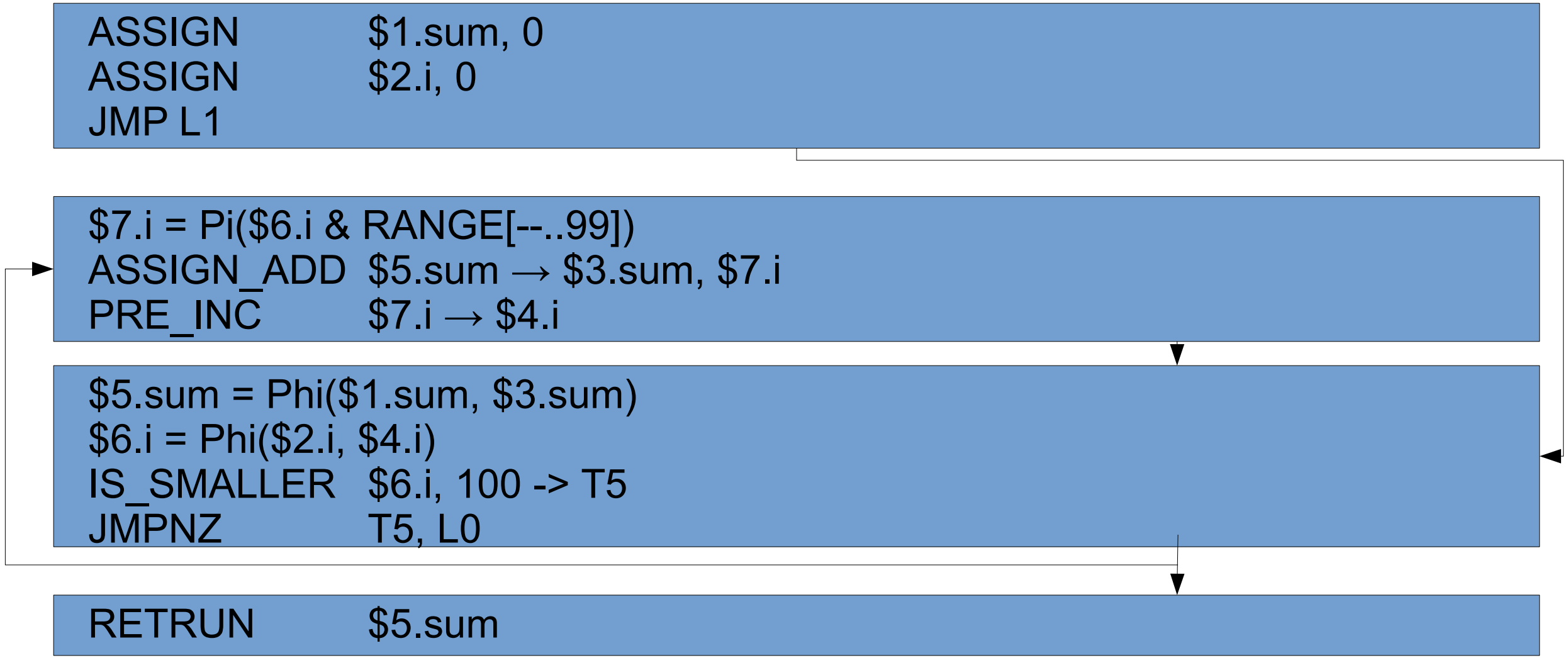


PHP 7.1 Optimizer (Extended Static Single Assignmnt Form)

```php
<?php
function sum() {
        $sum = 0;
        for ($i = 0; $i < 100; $i++) {
                $sum = $sum + $i;
        }
        return $sum;
}
```



PHP 7.1 Optimizer (Extended Static Single Assignmnt Form)



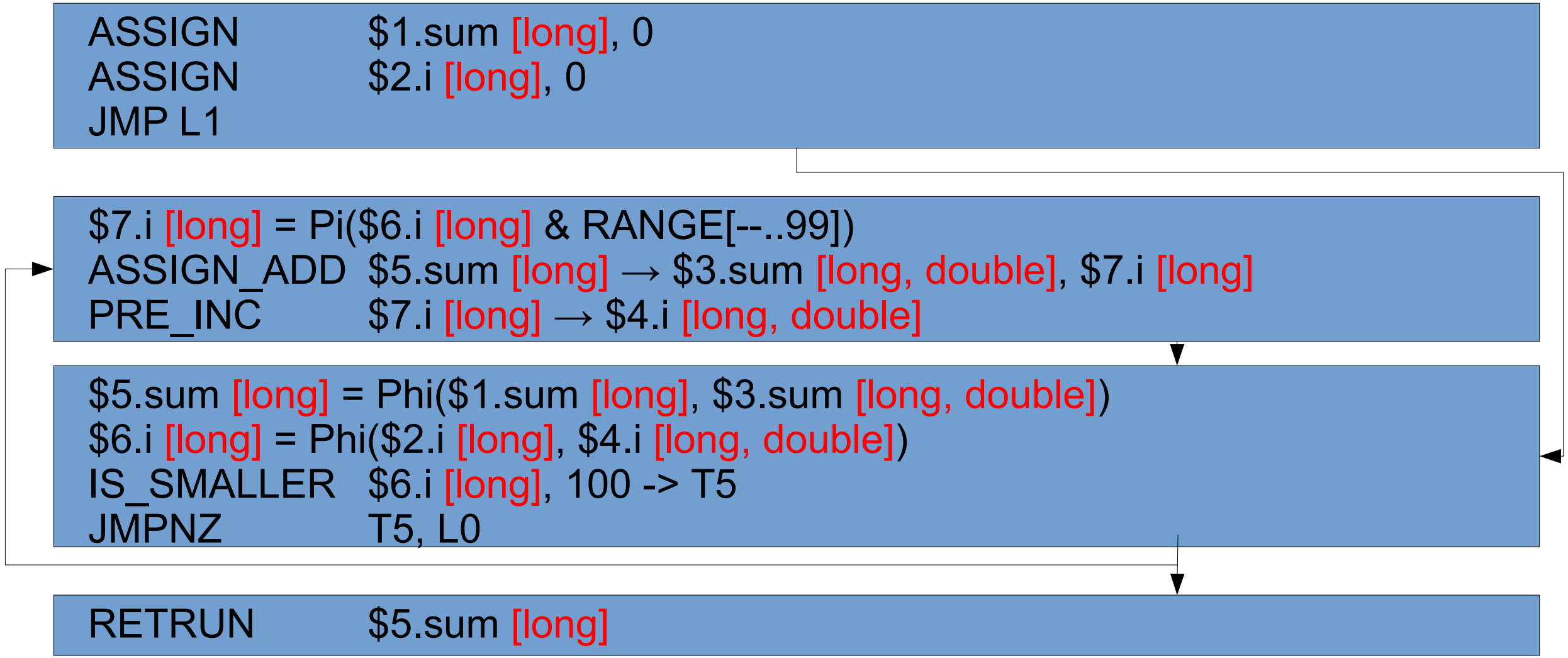
PHP 7.1 Optimizer (Type Propagation)

PHP 7.1 Optimizer (Type Propagation)

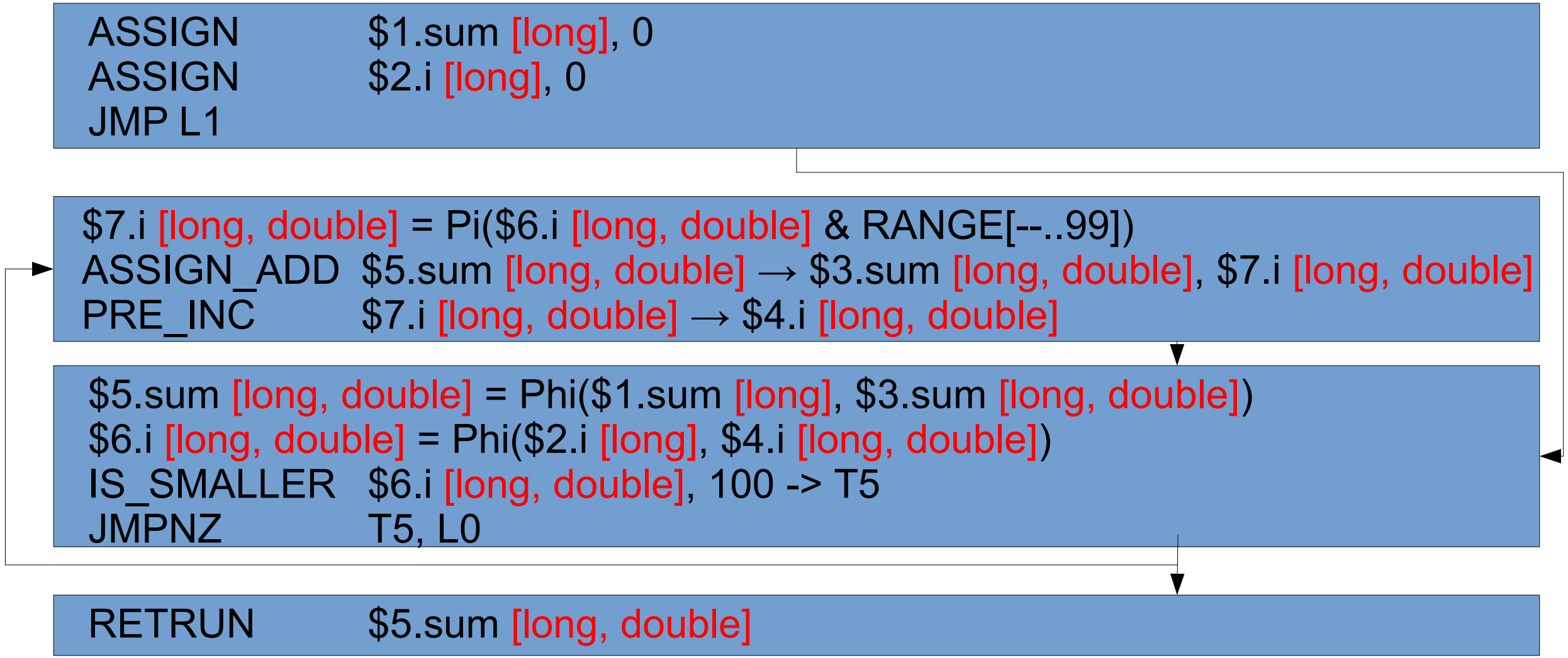
PHP 7.1 Optimizer (Type Propagation)

PHP 7.1 Optimizer (Type Propagation)

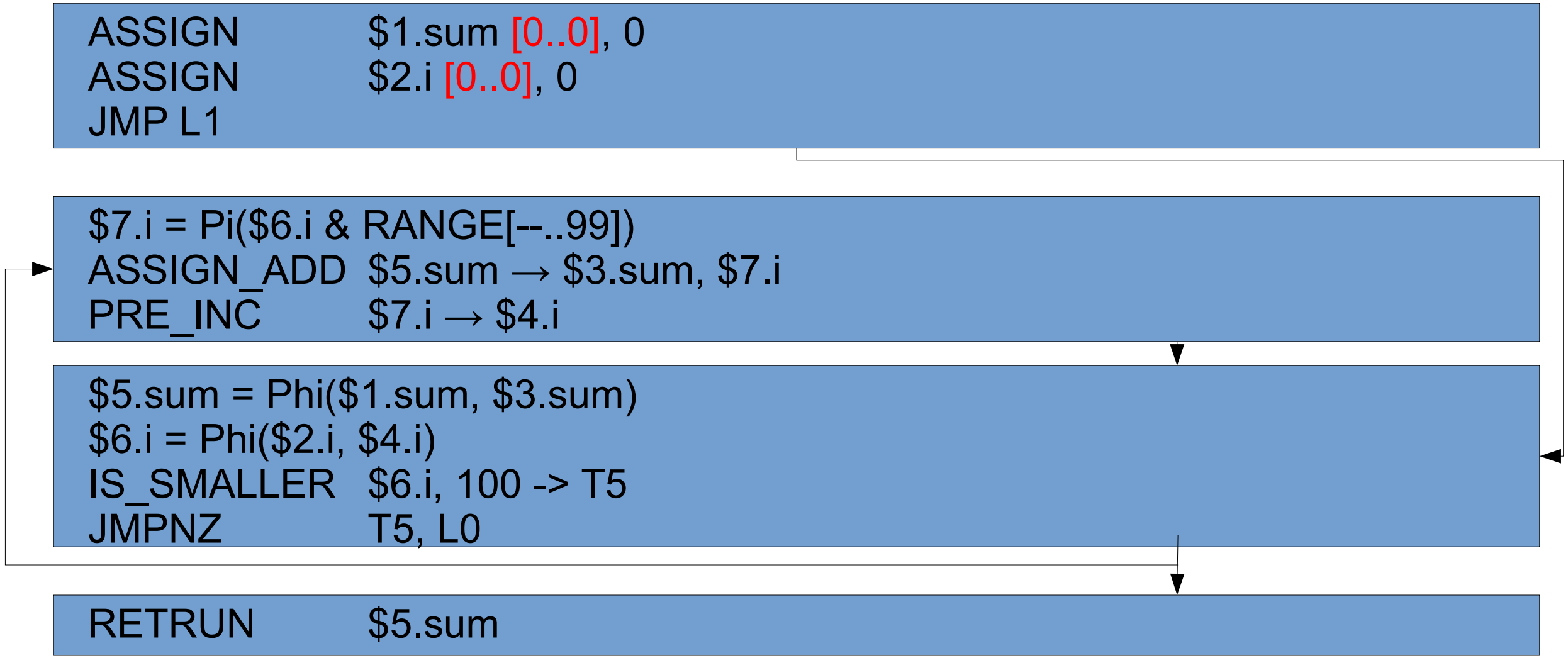
PHP 7.1 Optimizer (Type Propagation)



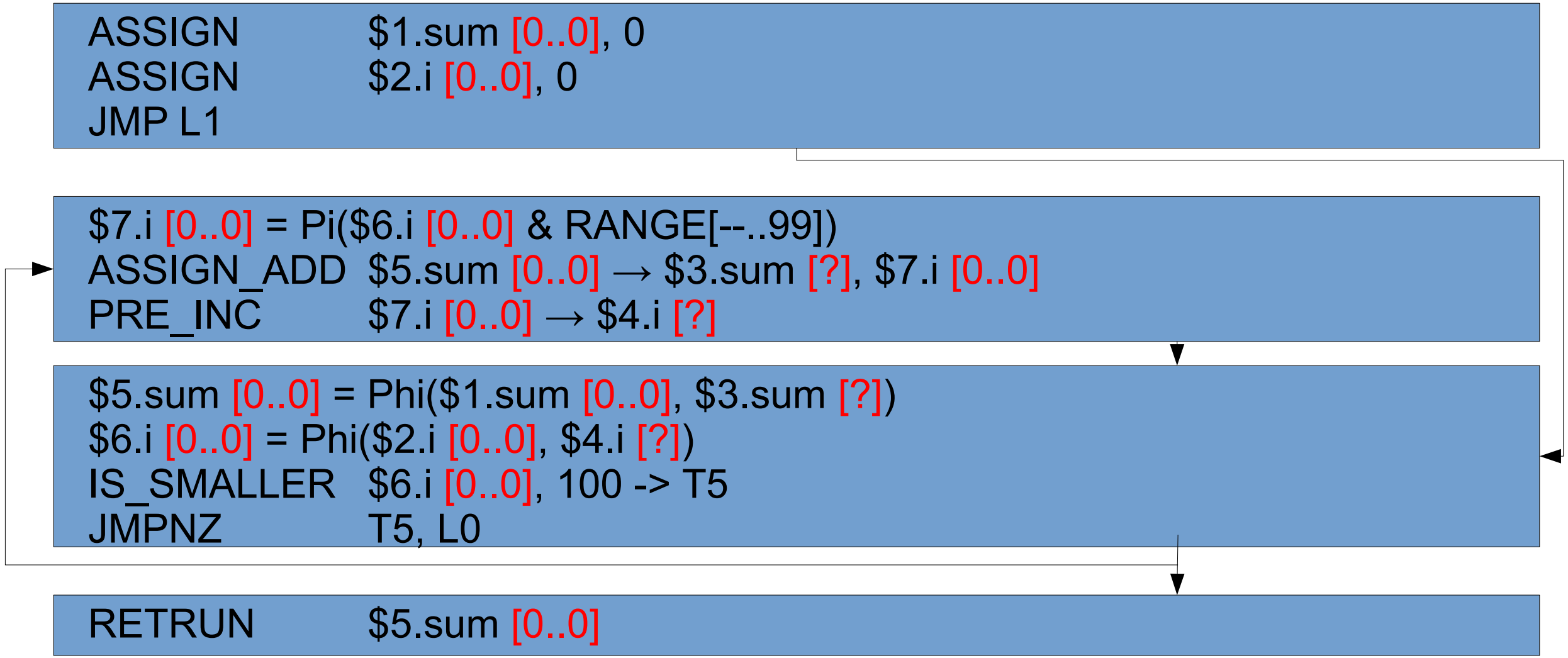
PHP 7.1 Optimizer (Type Propagation)



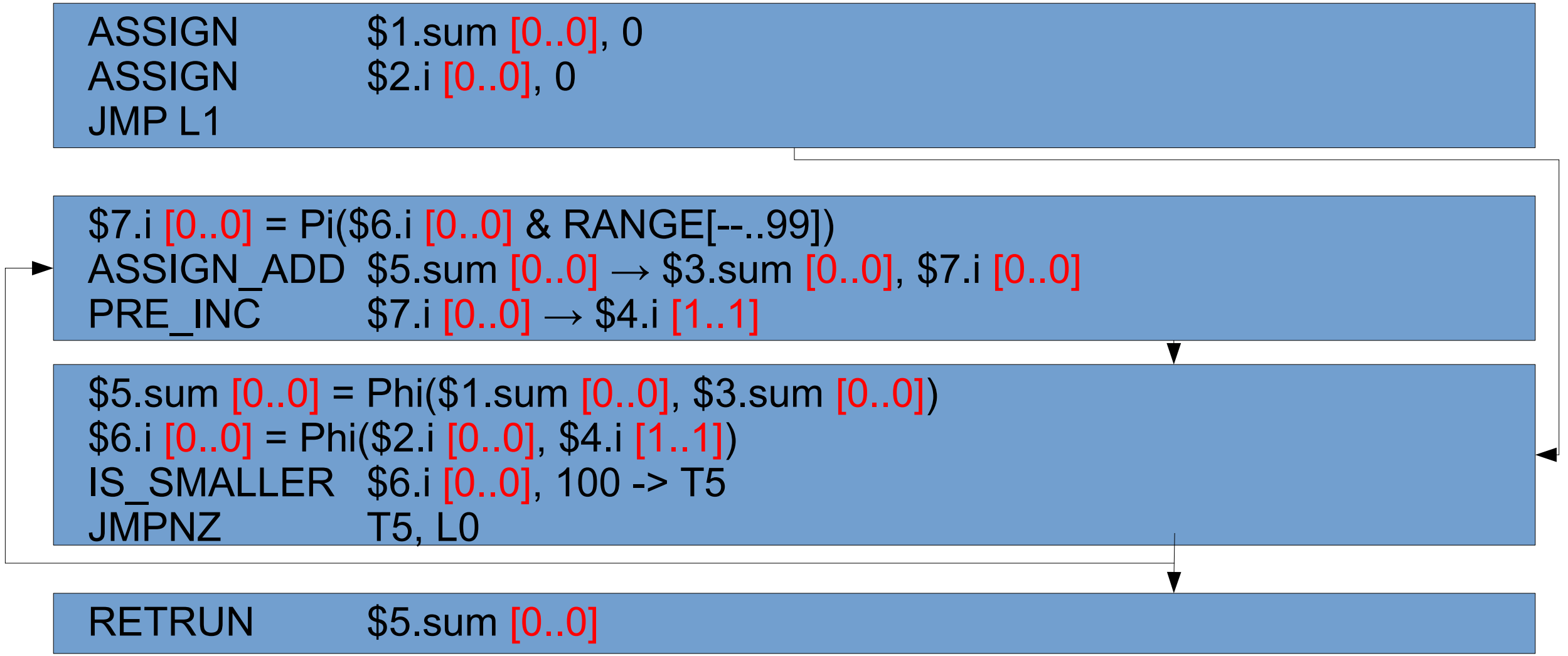
PHP 7.1 Optimizer (Range Propagation)



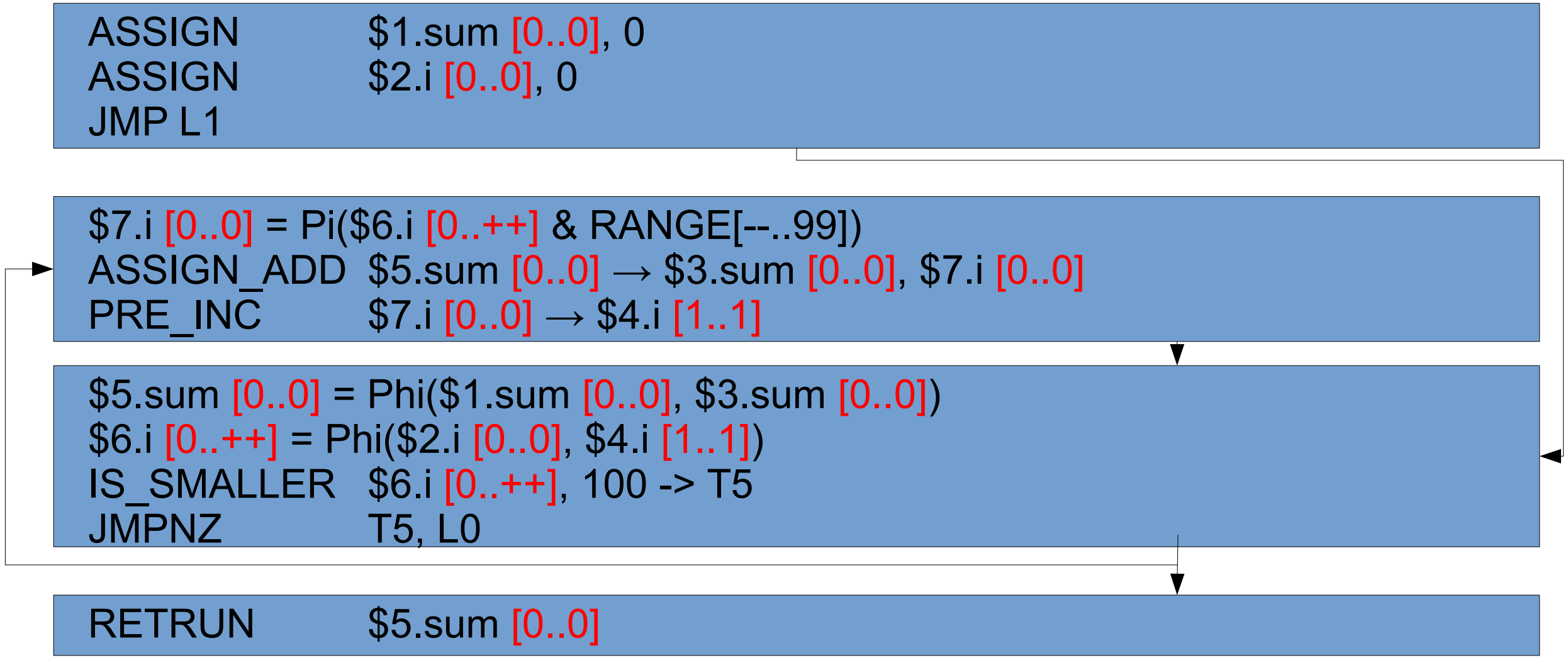
PHP 7.1 Optimizer (Range Propagation)



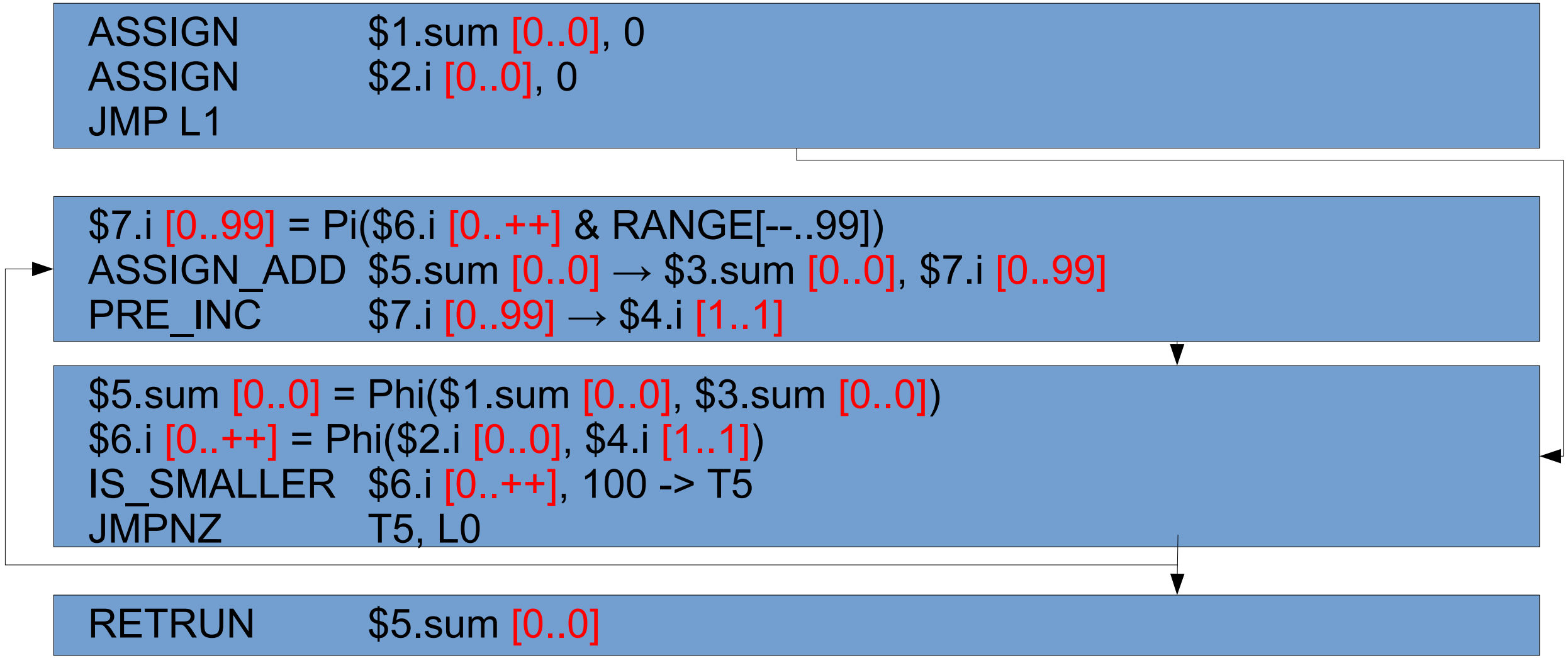
PHP 7.1 Optimizer (Range Propagation)



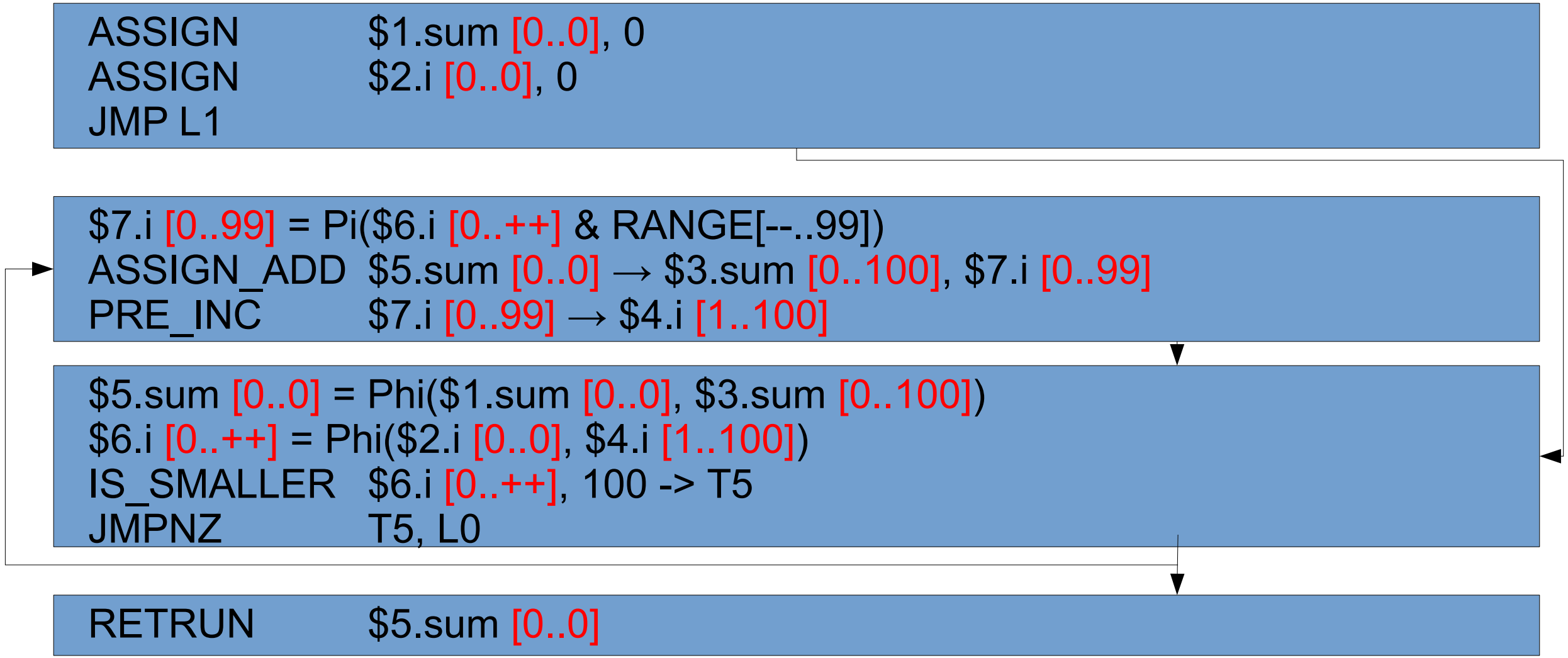
PHP 7.1 Optimizer (Range Propagation)



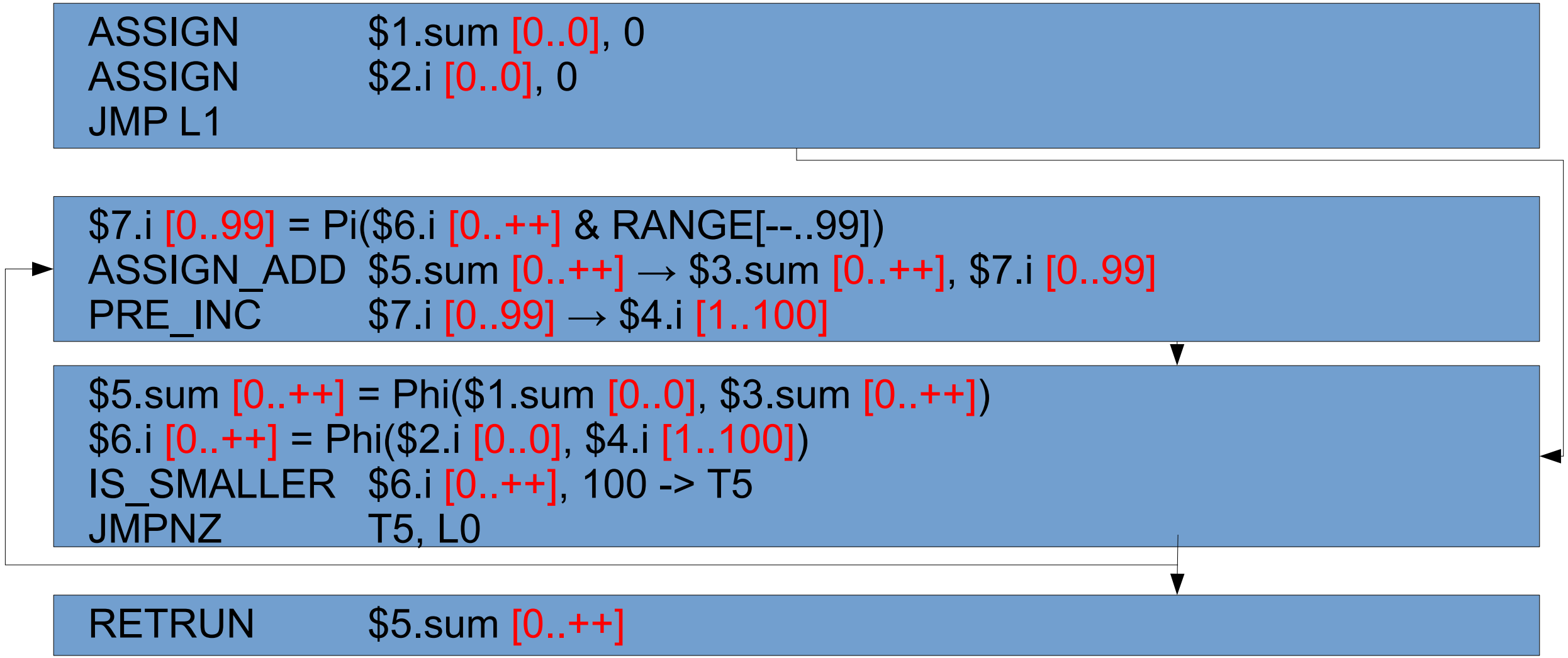
PHP 7.1 Optimizer (Range Propagation)



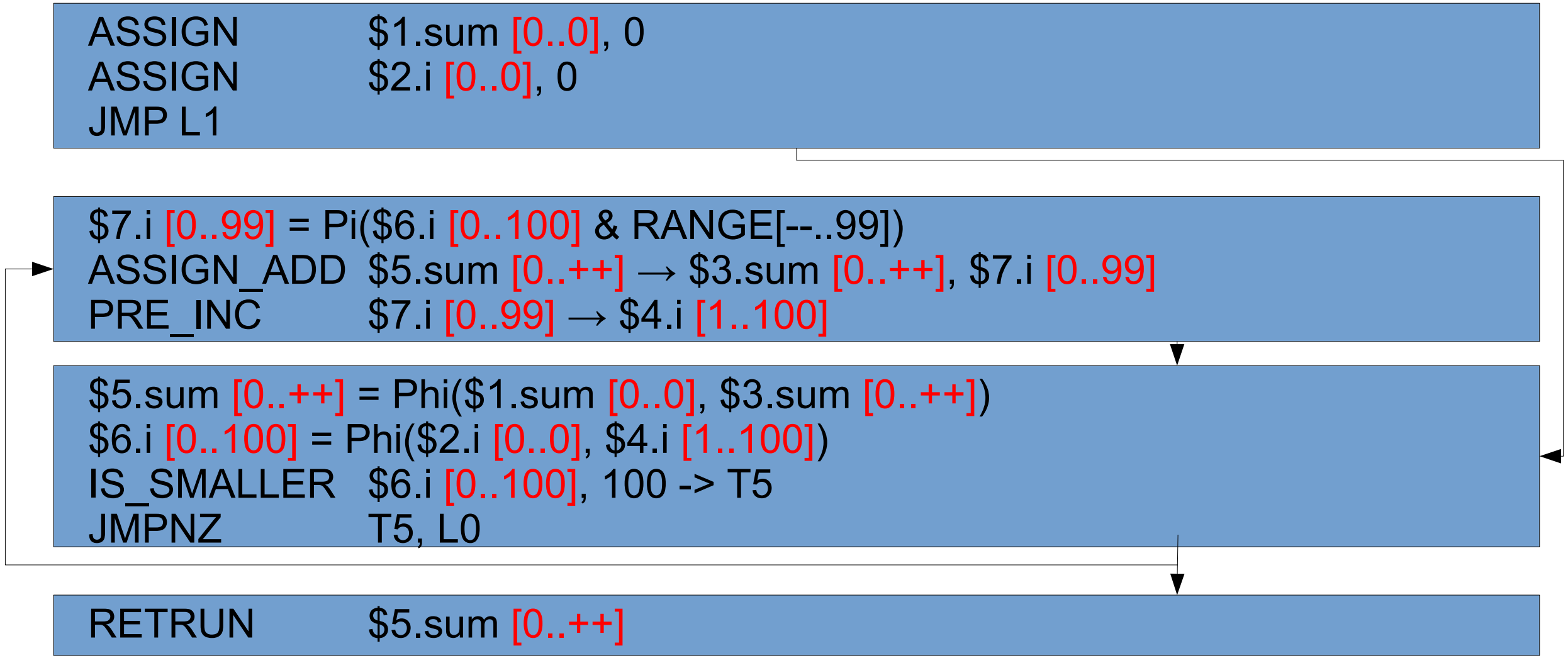
PHP 7.1 Optimizer (Range Propagation)



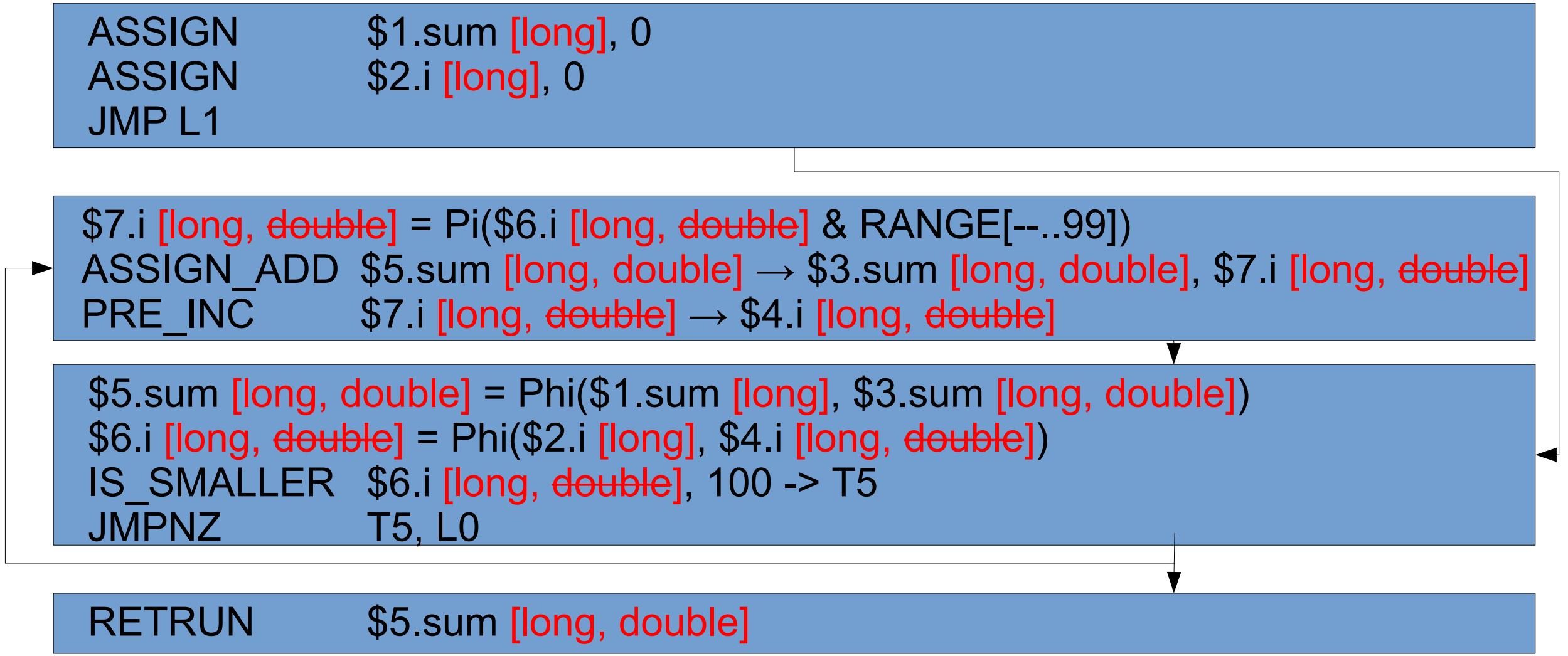
PHP 7.1 Optimizer (Range Propagation)



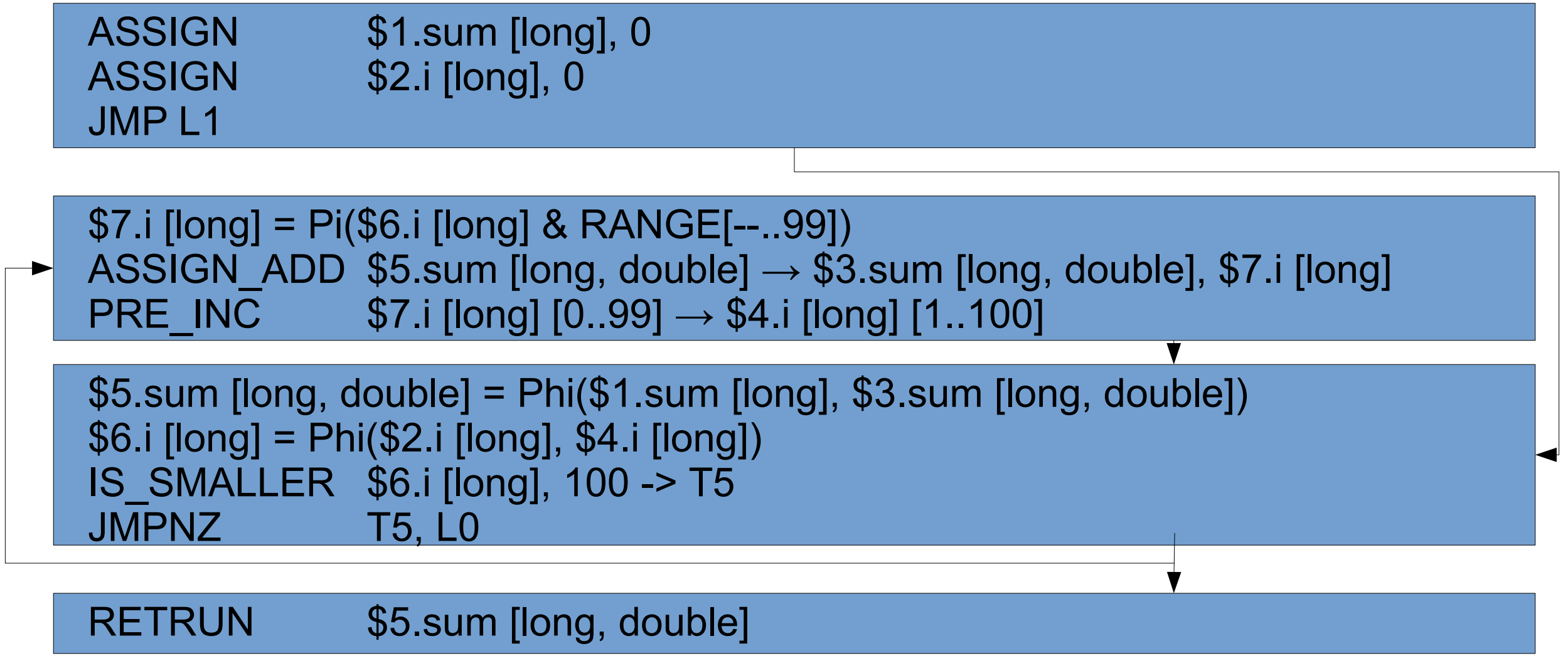
PHP 7.1 Optimizer (Range Propagation - narrowing)



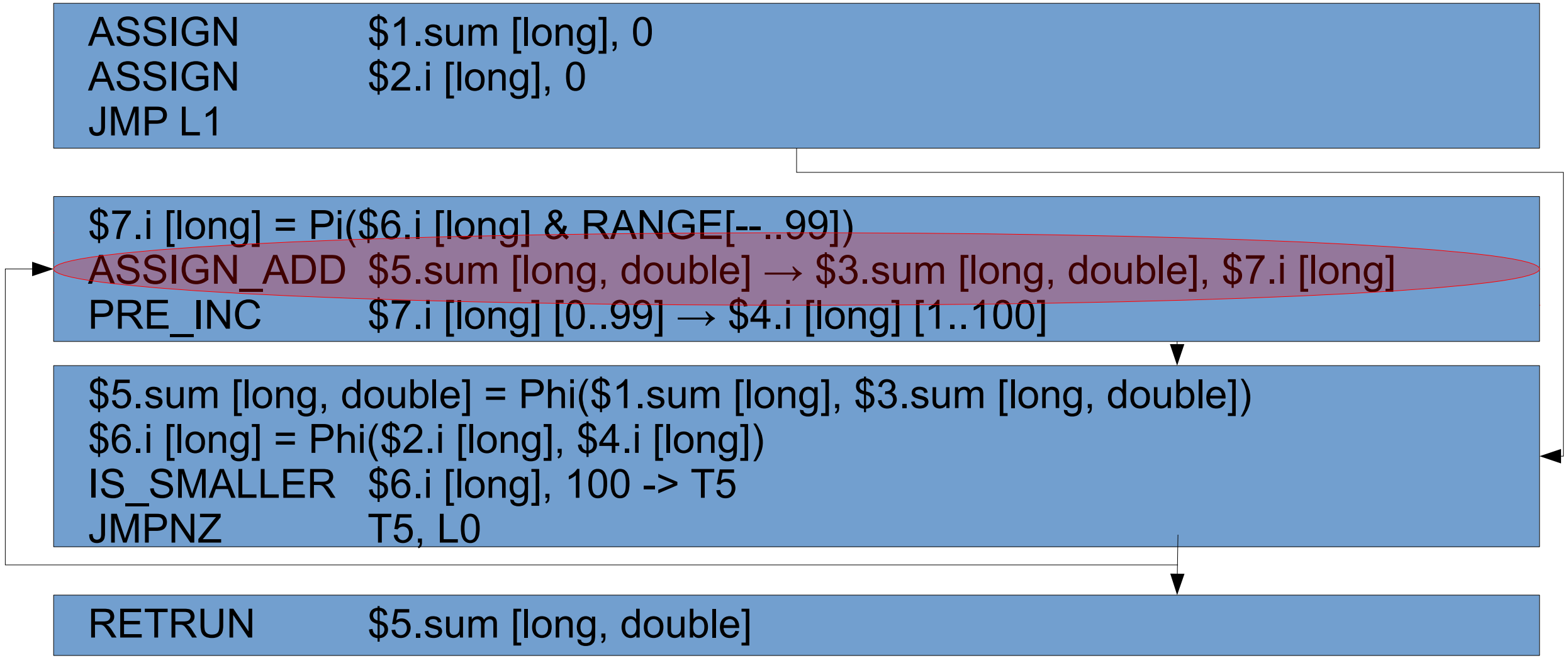
PHP 7.1 Optimizer (Range + Type Propagation)



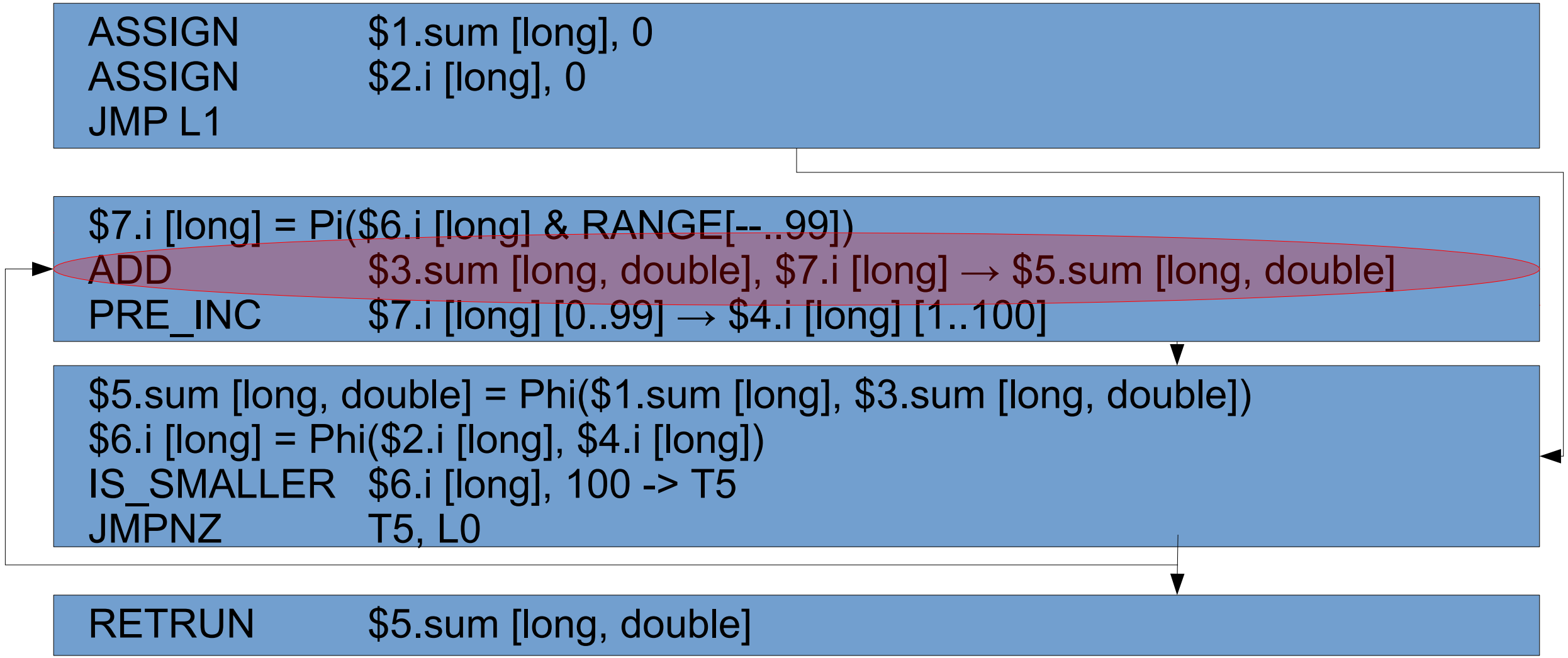
PHP 7.1 Optimizer (Range + Type Propagation)



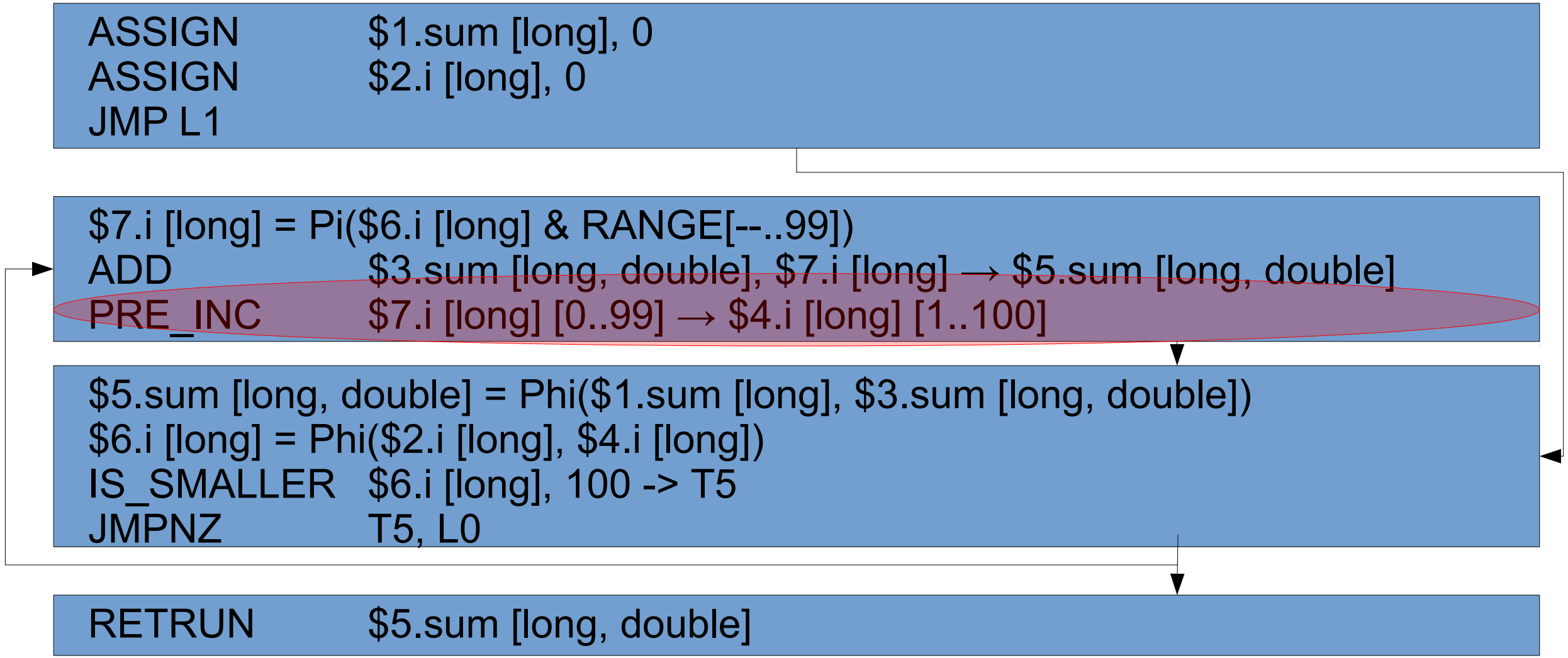
PHP 7.1 Optimizer (Optimization)



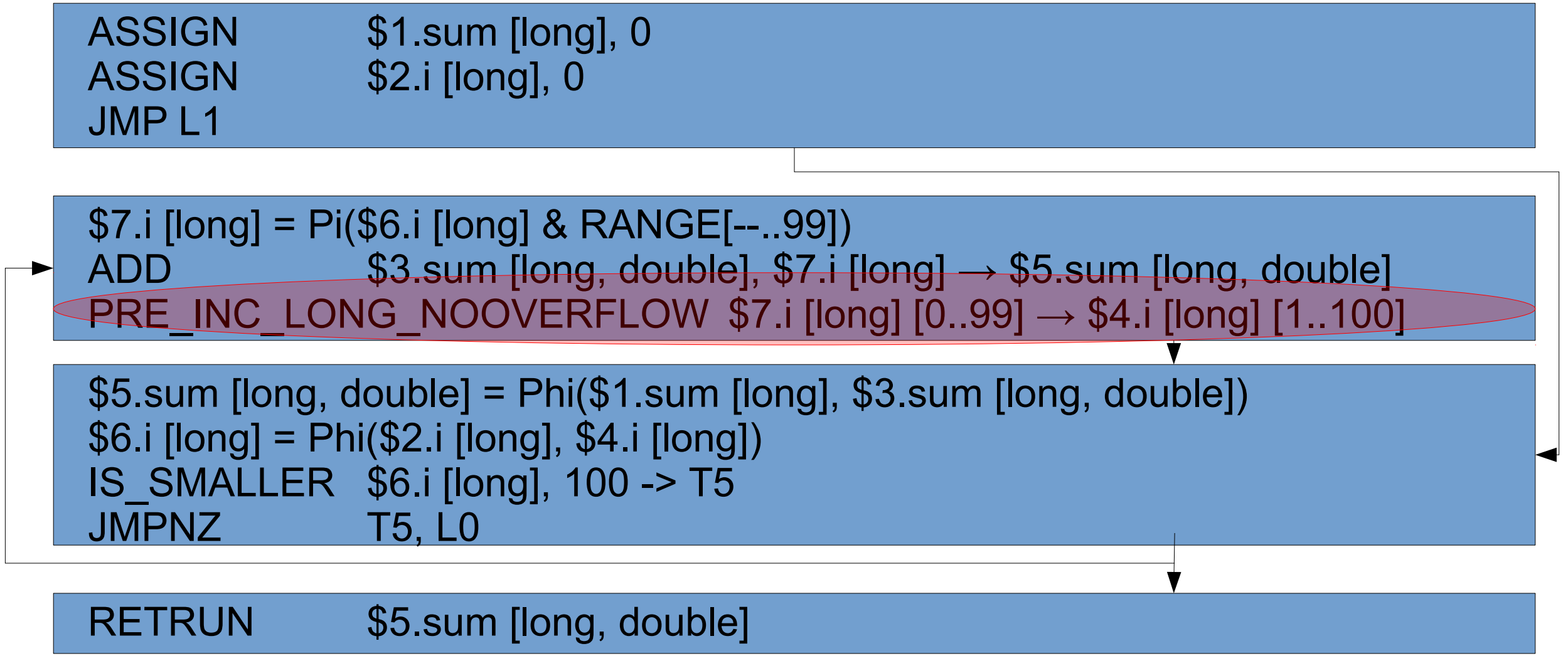
PHP 7.1 Optimizer (Optimization)



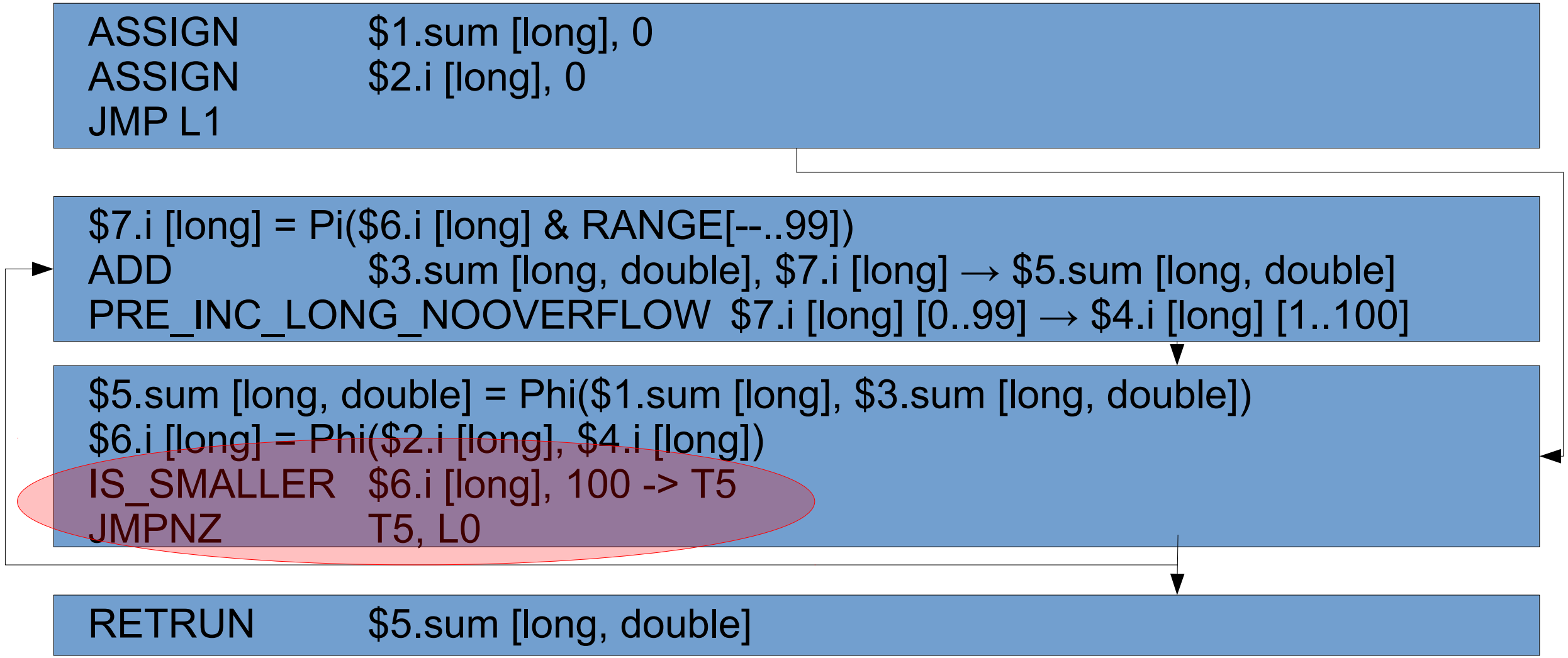
PHP 7.1 Optimizer (Optimization)



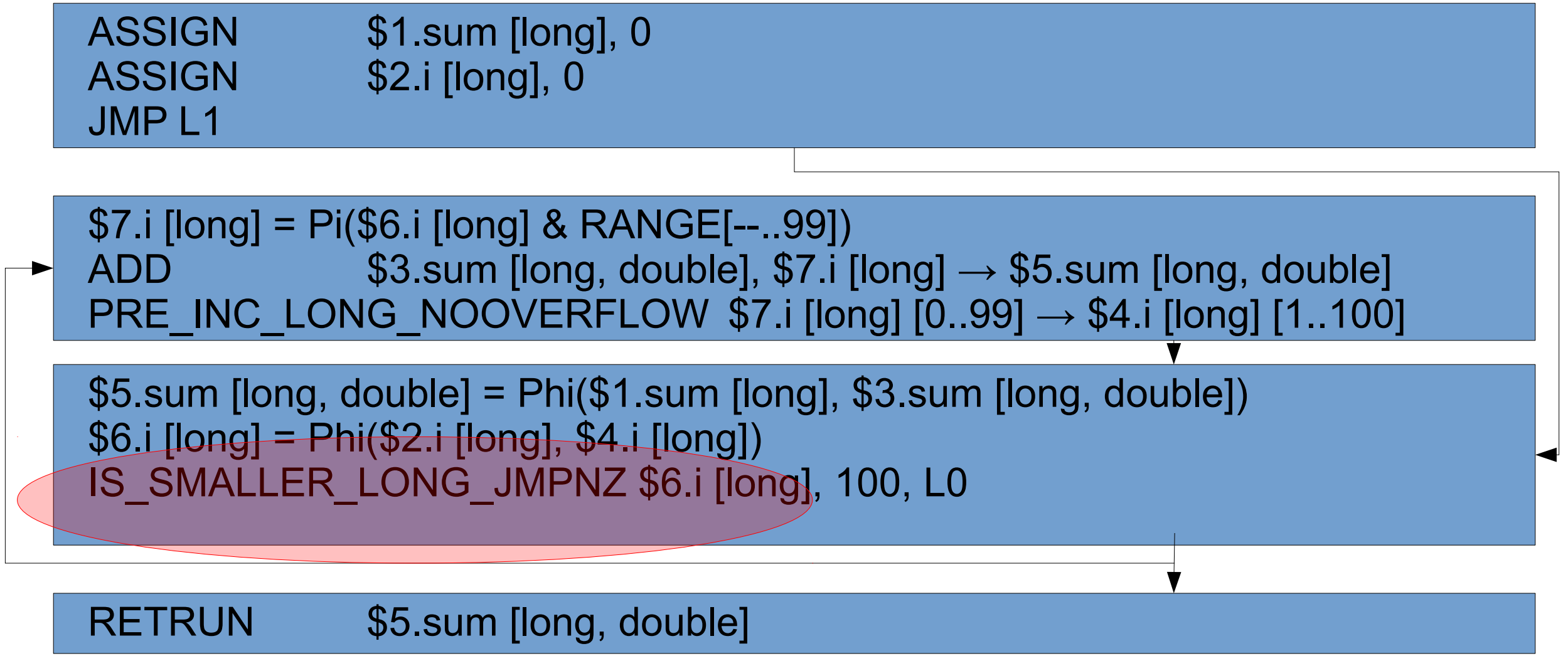
PHP 7.1 Optimizer (Optimization)



PHP 7.1 Optimizer (Optimization)



PHP 7.1 Optimizer (Optimization)



PHP 7.1 Optimizer (SSA deconstruction)

```
  ASSIGN          $sum, 0
  ASSIGN          $i, 0
  JMP L1
L0:
  ADD             $sum, $i -> T2
  ASSIGN          $sum, T2
  POST_INC        $i -> T4
  FREE            T4
L1:
  IS_SMALLER      $i, 100 -> T5
  JMPNZ           T5, L0
  RETRUN          $sum
  RETUTN          null
```

```
  ASSIGN          $sum, 0
  ASSIGN          $i, 0
  JMP L1
L0:
  ADD             $sum, $i  → $sum
  PRE_INC_LONG_NOOVERFLOW  $i
L1:
  IS_SMALLER_LONG_JMPNZ    $i, 100, L0
  RETRUN          $sum
```

PHP 7.1 Specialized Handlers

```
void PRE_INC_HANDLER()
{
    if (Z_TYPE_P(op1) != IS_LONG) {
        ... // not integer
    } else {
        Z_LVAL_P(op1)++;
        if (OVERFLOW) {
            ... // overflow
        }
    }
    CHECK_EXCEPTION();
    NEXT_OPCODE();
}
```

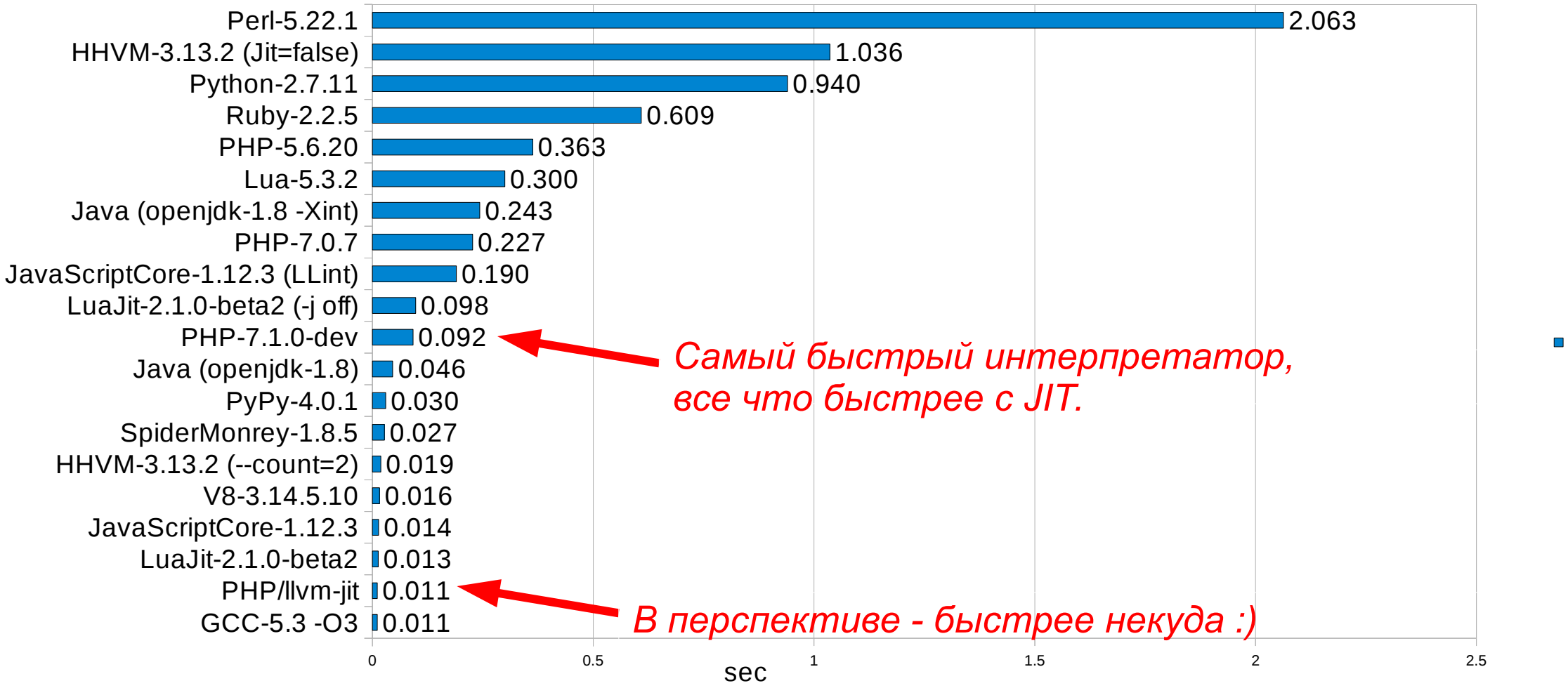
```
void
PRE_INC_HANDLER_LONG_NO_OVERFLOW()
{
        Z_LVAL_P(op1)++;
        NEXT_OPCODE();
}
```

```
mov 0x4(%IP), %eax    // get op1 offset
incl (%FP, %eax)      // increment
add 0x1c, %IP         // next opcode
ret
```

Что дальше?

- PHP 7.0
  - Оптимизация структур данных
- PHP 7.1
  - Анализатор потоков данных
  - Вывод типов
  - Глобальный оптимизатор для байт-кода PHP
  - Оптимизация и Специализация интерпретатора
- PHP 7.2
  - JIT?

The Computer Language Benchmarks Game (Mandelbrot)



Вопросы?

Dmitry Stogov

Principal Engineer at Zend Technologies

@dstogov

dmitry@zend.com

www.zend.com