MJIR Examples

```
class Main {
    public static void main(String[] args) {
      print(0);
    }
}
```

```
 # IR
 ('class', '@Main', None),
 ('define_void', '@Main.main', [('String[]', '%args')]),
 ('entry:',**negrito**),
 ('literal_int', 0, '%1'),
 ('print_int', '%1'),
 ('jump', '%exit'),
 ('exit:',),
 ('return_void',)

 # Pretty Print
 class @Main
 define void @Main.main (String[] %args)
 entry:
   %1 = literal int 0
   print int %1
   jump label %exit
 exit:
   return
```

```
class Main {
    public static void main(String[] args) {
        int i = -4;
        if(!(i>0)) {
            return;
        }
    }
}
```

```
 # IR
 ('class', '@Main', None),
 ('define_void', '@Main.main', [('String[]', '%args')]),
 ('entry:',),
 ('literal_int', 4, '%1'),
 ('literal_int', 0, '%2'),
 ('sub_int', '%2', '%1', '%3'),
 ('alloc_int', '%i'),
 ('store_int', '%3', '%i'),
 ('literal_int', 0, '%4'),
 ('load_int', '%i', '%5'),
 ('gt_int', '%5', '%4', '%6'),
 ('load_boolean', '%6', '%7'),
 ('not_boolean', '%7', '%8'),
 ('cbranch', '%8', '%if.then', '%if.end'),
 ('if.then:',),
 ('jump', '%exit'),
 ('if.end:',),
 ('jump', '%exit'),
 ('exit:',),
 ('return_void',)

 # Pretty Print
 class @Main
 define void @Main.main (String[] %args)
 entry:
   %1 = literal int 4
```

```
  %2 = literal int 0
  %3 = sub int %2 %1
  %i = alloc int
  store int %3 %i
  %4 = literal int 0
  %5 = load int %i
  %6 = gt int %5 %4
  %7 = load boolean %6
  %8 = not boolean %7
  cbranch %8 label %if.then label %if.end
 if.then:
  jump label %exit
 if.end:
  jump label %exit
 exit:
  return


class Main {
    int n = 3;

    public int doubleMe (int x) {
        return x * x;
    }

    public static void main(String[] args) {
        Main obj = new Main();
        int v = this.n;
        v = obj.doubleMe(v);
        assert v == this.n * this.n;
    }
}



 # IR
 ('global_String', '@.str.0', 'assertion_fail on 12:16'),
 ('class', '@Main', None),
 ('field_int', '@Main.n', 3),
 ('define_int', '@Main.doubleMe', [('int', '%1')]),
 ('entry:',),
 ('alloc_int', '%2'),
 ('alloc_int', '%x'),
 ('store_int', '%1', '%x'),
 ('load_int', '%x', '%3'),
 ('load_int', '%x', '%4'),
 ('mul_int', '%3', '%4', '%5'),
 ('store_int', '%5', '%2'),
 ('jump', '%exit'),
 ('exit:',),
 ('load_int', '%2', '%6'),
 ('return_int', '%6'),
 ('define_void', '@Main.main', [('String[]', '%args')]),
 ('entry:',),
 ('new_@Main', '%obj'),
 ('alloc_int', '%v'),
 ('store_int', '%this.n', '%v'),
 ('load_int', '%v', '%1'),
 ('param_int', '%1'),
 ('call_int', '%obj.doubleMe', '%2'),
 ('store_int', '%2', '%v'),
 ('load_int', '%this.n', '%3'),
 ('load_int', '%this.n', '%4'),
 ('mul_int', '%3', '%4', '%5'),
 ('load_int', '%v', '%6'),
 ('load_int', '%5', '%7'),
 ('eq_int', '%6', '%7', '%8'),
 ('cbranch', '%8', '%assert.true', '%assert.false'),
 ('assert.false:',),
 ('print_string', '@.str.0'),
 ('jump', '%exit'),
 ('assert.true:',),
 ('jump', '%exit'),
```

```
 ('exit:',),
 ('return_void',)

 # Pretty Print
 @.str.0 = global String 'assertion_fail on 12:16'

 class @Main
 field int @Main.n init = 3
 define int @Main.doubleMe (int %1)
 entry:
   %2 = alloc int
   %x = alloc int
   store int %1 %x
   %3 = load int %x
   %4 = load int %x
   %5 = mul int %3 %4
   store int %5 %2
   jump label %exit
 exit:
   %6 = load int %2
   return int %6
 define void @Main.main (String[] %args)
 entry:
   %obj = new @Main
   %v = alloc int
   store int %this.n %v
   %1 = load int %v
   param int %1
   %2 = call int %obj.doubleMe
   store int %2 %v
   %3 = load int %this.n
   %4 = load int %this.n
   %5 = mul int %3 %4
   %6 = load int %v
   %7 = load int %5
   %8 = eq int %6 %7
   cbranch %8 label %assert.true label %assert.false
 assert.false:
   print string @.str.0
   jump label %exit
 assert.true:
   jump label %exit
 exit:
   return


class Main {
    public static void main(String[] args) {
        int x, y = 3;
        int z = 4;
        x = y + z;
        z = -x;
        y = -z - z;
        assert y == 2*x;
    }
}



 # IR
 ('global_String', '@.str.0', 'assertion_fail on 8:16'),
 ('class', '@Main', None),
 ('define_void', '@Main.main', [('String[]', '%args')]),
 ('entry:',),
 ('alloc_int', '%x'),
 ('literal_int', 3, '%1'),
 ('alloc_int', '%y'),
 ('store_int', '%1', '%y'),
 ('literal_int', 4, '%2'),
 ('alloc_int', '%z'),
 ('store_int', '%2', '%z'),
 ('load_int', '%y', '%3'),
 ('load_int', '%z', '%4'),
```

```
('add_int', '%3', '%4', '%5'),
('store_int', '%5', '%x'),
('load_int', '%x', '%6'),
('literal_int', 0, '%7'),
('sub_int', '%7', '%6', '%8'),
('store_int', '%8', '%z'),
('load_int', '%z', '%9'),
('literal_int', 0, '%10'),
('sub_int', '%10', '%9', '%11'),
('load_int', '%11', '%12'),
('load_int', '%z', '%13'),
('sub_int', '%12', '%13', '%14'),
('store_int', '%14', '%y'),
('literal_int', 2, '%15'),
('load_int', '%15', '%16'),
('load_int', '%x', '%17'),
('mul_int', '%16', '%17', '%18'),
('load_int', '%y', '%19'),
('load_int', '%18', '%20'),
('eq_int', '%19', '%20', '%21'),
('cbranch', '%21', '%assert.true', '%assert.false'),
('assert.false:',),
('print_string', '@.str.0'),
('jump', '%exit'),
('assert.true:',),
('jump', '%exit'),
('exit:',),
('return_void',)

# Pretty Print
@.str.0 = global String 'assertion_fail on 8:16'

class @Main
define void @Main.main (String[] %args)
entry:
  %x = alloc int
  %1 = literal int 3
  %y = alloc int
  store int %1 %y
  %2 = literal int 4
  %z = alloc int
  store int %2 %z
  %3 = load int %y
  %4 = load int %z
  %5 = add int %3 %4
  store int %5 %x
  %6 = load int %x
  %7 = literal int 0
  %8 = sub int %7 %6
  store int %8 %z
  %9 = load int %z
  %10 = literal int 0
  %11 = sub int %10 %9
  %12 = load int %11
  %13 = load int %z
  %14 = sub int %12 %13
  store int %14 %y
  %15 = literal int 2
  %16 = load int %15
  %17 = load int %x
  %18 = mul int %16 %17
  %19 = load int %y
  %20 = load int %18
  %21 = eq int %19 %20
  cbranch %21 label %assert.true label %assert.false
assert.false:
  print string @.str.0
  jump label %exit
assert.true:
  jump label %exit
```

```
 exit:
   return


class Main {
    public static void main(String[] args) {
        int i, j, k = 0;
        i = 1;
        j = 2;
        for (int k=1; k<10; k = k+1)
            i = i + j * k;
        assert i == 91 && k == 0;
    }
}



 # IR
 ('global_String', '@.str.0', 'assertion_fail on 8:13'),
 ('class', '@Main', None),
 ('define_void', '@Main.main', [('String[]', '%args')]),
 ('entry:',),
 ('alloc_int', '%i'),
 ('alloc_int', '%j'),
 ('literal_int', 0, '%1'),
 ('alloc_int', '%k'),
 ('store_int', '%1', '%k'),
 ('literal_int', 1, '%2'),
 ('store_int', '%2', '%i'),
 ('literal_int', 2, '%3'),
 ('store_int', '%3', '%j'),
 ('literal_int', 1, '%4'),
 ('alloc_int', '%k.2'),
 ('store_int', '%4', '%k.2'),
 ('jump', '%for.cond'),
 ('for.cond:',),
 ('literal_int', 10, '%5'),
 ('load_int', '%k.2', '%6'),
 ('lt_int', '%6', '%5', '%7'),
 ('cbranch', '%7', '%for.body', '%for.end'),
 ('for.body:',),
 ('load_int', '%j', '%8'),
 ('load_int', '%k.2', '%9'),
 ('mul_int', '%8', '%9', '%10'),
 ('load_int', '%i', '%11'),
 ('load_int', '%10', '%12'),
 ('add_int', '%11', '%12', '%13'),
 ('store_int', '%13', '%i'),
 ('jump', '%for.inc'),
 ('for.inc:',),
 ('literal_int', 1, '%14'),
 ('load_int', '%k.2', '%15'),
 ('add_int', '%15', '%14', '%16'),
 ('store_int', '%16', '%k.2'),
 ('jump', '%for.cond'),
 ('for.end:',),
 ('literal_int', 91, '%17'),
 ('load_int', '%i', '%18'),
 ('eq_int', '%18', '%17', '%19'),
 ('literal_int', 0, '%20'),
 ('load_int', '%k', '%21'),
 ('eq_int', '%21', '%20', '%22'),
 ('load_boolean', '%19', '%23'),
 ('load_boolean', '%22', '%24'),
 ('and_boolean', '%23', '%24', '%25'),
 ('cbranch', '%25', '%assert.true', '%assert.false'),
 ('assert.false:',),
 ('print_string', '@.str.0'),
 ('jump', '%exit'),
 ('assert.true:',),
 ('jump', '%exit'),
 ('exit:',),
```

```
('return_void',)

# Pretty Print
@.str.0 = global String 'assertion_fail on 8:13'

class @Main
define void @Main.main (String[] %args)
entry:
  %i = alloc int
  %j = alloc int
  %1 = literal int 0
  %k = alloc int
  store int %1 %k
  %2 = literal int 1
  store int %2 %i
  %3 = literal int 2
  store int %3 %j
  %4 = literal int 1
  %k.2 = alloc int
  store int %4 %k.2
  jump label %for.cond
for.cond:
  %5 = literal int 10
  %6 = load int %k.2
  %7 = lt int %6 %5
  cbranch %7 label %for.body label %for.end
for.body:
  %8 = load int %j
  %9 = load int %k.2
  %10 = mul int %8 %9
  %11 = load int %i
  %12 = load int %10
  %13 = add int %11 %12
  store int %13 %i
  jump label %for.inc
for.inc:
  %14 = literal int 1
  %15 = load int %k.2
  %16 = add int %15 %14
  store int %16 %k.2
  jump label %for.cond
for.end:
  %17 = literal int 91
  %18 = load int %i
  %19 = eq int %18 %17
  %20 = literal int 0
  %21 = load int %k
  %22 = eq int %21 %20
  %23 = load boolean %19
  %24 = load boolean %22
  %25 = and boolean %23 %24
  cbranch %25 label %assert.true label %assert.false
assert.false:
  print string @.str.0
  jump label %exit
assert.true:
  jump label %exit
exit:
  return


class Main {
    int[] v = new int[5];
    public static void main(String[] args) {
        int[] f = {1, 2, 3, 4, 5};
        char[] s = "xpto";
        int i = 1, j = 0;
        print("Isto é um t: ", s[j+2]);
        print();
        this.v[j + i] = f[i];
        print(f[i], " == ", this.v[i]);
```

```
    }
}
```

```
# IR
('global_int[]_5', '@.const_f.0', [1, 2, 3, 4, 5]),
('global_String', '@.str.1', 'xpto'),
('global_String', '@.str.2', 'Isto é um t: '),
('global_String', '@.str.3', ' == '),
('class', '@Main', None),
('field_int[]', '@Main.v', 5),
('define_void', '@Main.main', [('String[]', '%args')]),
('entry:',),
('alloc_int[]_5', '%f'),
('store_int[]_5', '@.const_f.0', '%f'),
('alloc_char[]_4', '%s'),
('store_char[]_4', '@.str.1', '%s'),
('literal_int', 1, '%1'),
('alloc_int', '%i'),
('store_int', '%1', '%i'),
('literal_int', 0, '%2'),
('alloc_int', '%j'),
('store_int', '%2', '%j'),
('literal_int', 2, '%3'),
('load_int', '%j', '%4'),
('add_int', '%4', '%3', '%5'),
('elem_char', '%s', '%5', '%6'),
('print_String', '@.str.2'),
('load_char_*', '%6', '%7'),
('print_char', '%7'),
('print_void',),
('load_int', '%j', '%8'),
('load_int', '%i', '%9'),
('add_int', '%8', '%9', '%10'),
('elem_int', '%this.v', '%10', '%11'),
('elem_int', '%f', '%i', '%12'),
('load_int_*', '%12', '%13'),
('store_int_*', '%13', '%11'),
('elem_int', '%f', '%i', '%14'),
('elem_int', '%this.v', '%i', '%15'),
('load_int_*', '%14', '%16'),
('print_int', '%16'),
('print_String', '@.str.3'),
('load_int_*', '%15', '%17'),
('print_int', '%17'),
('jump', '%exit'),
('exit:',),
('return_void',)

# Pretty Print
@.const_f.0 = global int[][5] [1, 2, 3, 4, 5]
@.str.1 = global String 'xpto'
@.str.2 = global String 'Isto é um t: '
@.str.3 = global String ' == '

class @Main
field int[] @Main.v init = 5
define void @Main.main (String[] %args)
entry:
  %f = alloc int[][5]
  store int[][5] @.const_f.0 %f
  %s = alloc char[][4]
  store char[][4] @.str.1 %s
  %1 = literal int 1
  %i = alloc int
  store int %1 %i
  %2 = literal int 0
  %j = alloc int
  store int %2 %j
  %3 = literal int 2
```

```
  %4 = load int %j
  %5 = add int %4 %3
  %6 = elem char %s %5
  print String @.str.2
  %7 = load char* %6
  print char %7
  print
  %8 = load int %j
  %9 = load int %i
  %10 = add int %8 %9
  %11 = elem int %this.v %10
  %12 = elem int %f %i
  %13 = load int* %12
  store int* %13 %11
  %14 = elem int %f %i
  %15 = elem int %this.v %i
  %16 = load int* %14
  print int %16
  print String @.str.3
  %17 = load int* %15
  print int %17
  jump label %exit
exit:
  return


/* Bubble sort code */

class Main {
    public static void main(String[] args) {
        int[] v = new int[100];
        int n, c, d, swap;
        // Number of elements:
        n = 25;
        for (c = 0; c < n; c = c+1)
            v[c] = n - c;
        for (c = 0; c < n-1; c = c+1)
            for (d = 0; d < n-c-1; d = d+1)
                if (v[d] > v[d+1]) {
                    swap = v[d];
                    v[d] = v[d+1];
                    v[d+1] = swap;
                }
        print("Sorted list in ascending order: ");
        for (c = 0; c < n; c = c+1)
            print(v[c], " ");
    }
}


 # IR
 ('global_String', '@.str.0', 'Sorted list in ascending order: '),
 ('global_String', '@.str.1', ' '),
 ('class', '@Main', None),
 ('define_void', '@Main.main', [('String[]', '%args')]),
 ('entry:',),
 ('new_int[]_100', '%v'),
 ('alloc_int', '%n'),
 ('alloc_int', '%c'),
 ('alloc_int', '%d'),
 ('alloc_int', '%swap'),
 ('literal_int', 25, '%1'),
 ('store_int', '%1', '%n'),
 ('literal_int', 0, '%2'),
 ('store_int', '%2', '%c'),
 ('jump', '%for.cond'),
 ('for.cond:',),
 ('load_int', '%c', '%3'),
 ('load_int', '%n', '%4'),
 ('lt_int', '%3', '%4', '%5'),
 ('cbranch', '%5', '%for.body', '%for.end'),
 ('for.body:',),
 ('elem_int', '%v', '%c', '%6'),
```

```
('load_int', '%n', '%7'),
('load_int', '%c', '%8'),
('sub_int', '%7', '%8', '%9'),
('store_int_*', '%9', '%6'),
('jump', '%for.inc'),
('for.inc:',),
('literal_int', 1, '%10'),
('load_int', '%c', '%11'),
('add_int', '%11', '%10', '%12'),
('store_int', '%12', '%c'),
('jump', '%for.cond'),
('for.end:',),
('literal_int', 0, '%13'),
('store_int', '%13', '%c'),
('jump', '%for.cond.1'),
('for.cond.1:',),
('literal_int', 1, '%14'),
('load_int', '%n', '%15'),
('sub_int', '%15', '%14', '%16'),
('load_int', '%c', '%17'),
('load_int', '%16', '%18'),
('lt_int', '%17', '%18', '%19'),
('cbranch', '%19', '%for.body.1', '%for.end.1'),
('for.body.1:',),
('literal_int', 0, '%20'),
('store_int', '%20', '%d'),
('jump', '%for.cond.2'),
('for.cond.2:',),
('load_int', '%n', '%21'),
('load_int', '%c', '%22'),
('sub_int', '%21', '%22', '%23'),
('literal_int', 1, '%24'),
('load_int', '%23', '%25'),
('sub_int', '%25', '%24', '%26'),
('load_int', '%d', '%27'),
('load_int', '%26', '%28'),
('lt_int', '%27', '%28', '%29'),
('cbranch', '%29', '%for.body.2', '%for.end.2'),
('for.body.2:',),
('elem_int', '%v', '%d', '%30'),
('literal_int', 1, '%31'),
('load_int', '%d', '%32'),
('add_int', '%32', '%31', '%33'),
('elem_int', '%v', '%33', '%34'),
('load_int_*', '%30', '%35'),
('load_int_*', '%34', '%36'),
('gt_int', '%35', '%36', '%37'),
('cbranch', '%37', '%if.then', '%if.end'),
('if.then:',),
('elem_int', '%v', '%d', '%38'),
('load_int_*', '%38', '%39'),
('store_int', '%39', '%swap'),
('elem_int', '%v', '%d', '%40'),
('literal_int', 1, '%41'),
('load_int', '%d', '%42'),
('add_int', '%42', '%41', '%43'),
('elem_int', '%v', '%43', '%44'),
('load_int_*', '%44', '%45'),
('store_int_*', '%45', '%40'),
('literal_int', 1, '%46'),
('load_int', '%d', '%47'),
('add_int', '%47', '%46', '%48'),
('elem_int', '%v', '%48', '%49'),
('store_int_*', '%swap', '%49'),
('jump', '%if.end'),
('if.end:',),
('jump', '%for.inc.2'),
('for.inc.2:',),
('literal_int', 1, '%50'),
('load_int', '%d', '%51'),
```

```
('add_int', '%51', '%50', '%52'),
('store_int', '%52', '%d'),
('jump', '%for.cond.2'),
('for.end.2:',),
('jump', '%for.inc.1'),
('for.inc.1:',),
('literal_int', 1, '%53'),
('load_int', '%c', '%54'),
('add_int', '%54', '%53', '%55'),
('store_int', '%55', '%c'),
('jump', '%for.cond.1'),
('for.end.1:',),
('print_String', '@.str.0'),
('literal_int', 0, '%56'),
('store_int', '%56', '%c'),
('jump', '%for.cond.3'),
('for.cond.3:',),
('load_int', '%c', '%57'),
('load_int', '%n', '%58'),
('lt_int', '%57', '%58', '%59'),
('cbranch', '%59', '%for.body.3', '%for.end.3'),
('for.body.3:',),
('elem_int', '%v', '%c', '%60'),
('load_int_*', '%60', '%61'),
('print_int', '%61'),
('print_String', '@.str.1'),
('jump', '%for.inc.3'),
('for.inc.3:',),
('literal_int', 1, '%62'),
('load_int', '%c', '%63'),
('add_int', '%63', '%62', '%64'),
('store_int', '%64', '%c'),
('jump', '%for.cond.3'),
('for.end.3:',),
('jump', '%exit'),
('exit:',),
('return_void',)

# Pretty Print
@.str.0 = global String 'Sorted list in ascending order: '
@.str.1 = global String ' '

class @Main
define void @Main.main (String[] %args)
entry:
  %v = new int[][100]
  %n = alloc int
  %c = alloc int
  %d = alloc int
  %swap = alloc int
  %1 = literal int 25
  store int %1 %n
  %2 = literal int 0
  store int %2 %c
  jump label %for.cond
for.cond:
  %3 = load int %c
  %4 = load int %n
  %5 = lt int %3 %4
  cbranch %5 label %for.body label %for.end
for.body:
  %6 = elem int %v %c
  %7 = load int %n
  %8 = load int %c
  %9 = sub int %7 %8
  store int* %9 %6
  jump label %for.inc
for.inc:
  %10 = literal int 1
  %11 = load int %c
```

```
  %12 = add int %11 %10
  store int %12 %c
  jump label %for.cond
for.end:
  %13 = literal int 0
  store int %13 %c
  jump label %for.cond.1
for.cond.1:
  %14 = literal int 1
  %15 = load int %n
  %16 = sub int %15 %14
  %17 = load int %c
  %18 = load int %16
  %19 = lt int %17 %18
  cbranch %19 label %for.body.1 label %for.end.1
for.body.1:
  %20 = literal int 0
  store int %20 %d
  jump label %for.cond.2
for.cond.2:
  %21 = load int %n
  %22 = load int %c
  %23 = sub int %21 %22
  %24 = literal int 1
  %25 = load int %23
  %26 = sub int %25 %24
  %27 = load int %d
  %28 = load int %26
  %29 = lt int %27 %28
  cbranch %29 label %for.body.2 label %for.end.2
for.body.2:
  %30 = elem int %v %d
  %31 = literal int 1
  %32 = load int %d
  %33 = add int %32 %31
  %34 = elem int %v %33
  %35 = load int* %30
  %36 = load int* %34
  %37 = gt int %35 %36
  cbranch %37 label %if.then label %if.end
if.then:
  %38 = elem int %v %d
  %39 = load int* %38
  store int %39 %swap
  %40 = elem int %v %d
  %41 = literal int 1
  %42 = load int %d
  %43 = add int %42 %41
  %44 = elem int %v %43
  %45 = load int* %44
  store int* %45 %40
  %46 = literal int 1
  %47 = load int %d
  %48 = add int %47 %46
  %49 = elem int %v %48
  store int* %swap %49
  jump label %if.end
if.end:
  jump label %for.inc.2
for.inc.2:
  %50 = literal int 1
  %51 = load int %d
  %52 = add int %51 %50
  store int %52 %d
  jump label %for.cond.2
for.end.2:
  jump label %for.inc.1
for.inc.1:
  %53 = literal int 1
  %54 = load int %c
```

```
  %55 = add int %54 %53
  store int %55 %c
  jump label %for.cond.1
 for.end.1:
  print String @.str.0
  %56 = literal int 0
  store int %56 %c
  jump label %for.cond.3
 for.cond.3:
  %57 = load int %c
  %58 = load int %n
  %59 = lt int %57 %58
  cbranch %59 label %for.body.3 label %for.end.3
 for.body.3:
  %60 = elem int %v %c
  %61 = load int* %60
  print int %61
  print String @.str.1
  jump label %for.inc.3
 for.inc.3:
  %62 = literal int 1
  %63 = load int %c
  %64 = add int %63 %62
  store int %64 %c
  jump label %for.cond.3
 for.end.3:
  jump label %exit
 exit:
  return
```

```
class Arithmetic {
    public int add(int x, int y) {
        return x + y;
    }

    public int subtract(int x, int y) {
        return x - y;
    }
}

class Main {
    public static void main(String[] args) {
       int  foo, bar;
       Arithmetic arith = new Arithmetic();
       foo = 5;
       bar = 15;
       print(foo, " + ", bar, " = ", arith.add(foo, bar), ", ");
       print(foo, " - ", bar, " = ", arith.subtract(foo, bar));
   }
}
```

```
 # IR
 ('global_String', '@.str.0', ' + '),
 ('global_String', '@.str.1', ' = '),
 ('global_String', '@.str.2', ', '),
 ('global_String', '@.str.3', ' - '),
 ('global_String', '@.str.4', ' = '),
 ('class', '@Arithmetic', None),
 ('define_int', '@Arithmetic.add', [('int', '%1'), ('int', '%2')]),
 ('entry:',),
 ('alloc_int', '%3'),
 ('alloc_int', '%x'),
 ('alloc_int', '%y'),
 ('store_int', '%1', '%x'),
 ('store_int', '%2', '%y'),
 ('load_int', '%x', '%4'),
 ('load_int', '%y', '%5'),
 ('add_int', '%4', '%5', '%6'),
 ('store_int', '%6', '%3'),
 ('jump', '%exit'),
 ('exit:',),
```

```
('load_int', '%3', '%7'),
('return_int', '%7'),
('define_int', '@Arithmetic.subtract', [('int', '%1'), ('int', '%2')]),
('entry:',),
('alloc_int', '%3'),
('alloc_int', '%x'),
('alloc_int', '%y'),
('store_int', '%1', '%x'),
('store_int', '%2', '%y'),
('load_int', '%x', '%4'),
('load_int', '%y', '%5'),
('sub_int', '%4', '%5', '%6'),
('store_int', '%6', '%3'),
('jump', '%exit'),
('exit:',),
('load_int', '%3', '%7'),
('return_int', '%7'),
('class', '@Main', None),
('define_void', '@Main.main', [('String[]', '%args')]),
('entry:',),
('alloc_int', '%foo'),
('alloc_int', '%bar'),
('new_@Arithmetic', '%arith'),
('literal_int', 5, '%1'),
('store_int', '%1', '%foo'),
('literal_int', 15, '%2'),
('store_int', '%2', '%bar'),
('load_int', '%foo', '%3'),
('param_int', '%3'),
('load_int', '%bar', '%4'),
('param_int', '%4'),
('call_int', '%arith.add', '%5'),
('load_int', '%foo', '%6'),
('print_int', '%6'),
('print_String', '@.str.0'),
('load_int', '%bar', '%7'),
('print_int', '%7'),
('print_String', '@.str.1'),
('print_int', '%5'),
('print_String', '@.str.2'),
('load_int', '%foo', '%8'),
('param_int', '%8'),
('load_int', '%bar', '%9'),
('param_int', '%9'),
('call_int', '%arith.subtract', '%10'),
('load_int', '%foo', '%11'),
('print_int', '%11'),
('print_String', '@.str.3'),
('load_int', '%bar', '%12'),
('print_int', '%12'),
('print_String', '@.str.4'),
('print_int', '%10'),
('jump', '%exit'),
('exit:',),
('return_void',)

# Pretty Print
@.str.0 = global String ' + '
@.str.1 = global String ' = '
@.str.2 = global String ', '
@.str.3 = global String ' - '
@.str.4 = global String ' = '

class @Arithmetic
define int @Arithmetic.add (int %1, int %2)
entry:
  %3 = alloc int
  %x = alloc int
  %y = alloc int
  store int %1 %x
```

```
  store int %2 %y
  %4 = load int %x
  %5 = load int %y
  %6 = add int %4 %5
  store int %6 %3
  jump label %exit
exit:
  %7 = load int %3
  return int %7
define int @Arithmetic.subtract (int %1, int %2)
entry:
  %3 = alloc int
  %x = alloc int
  %y = alloc int
  store int %1 %x
  store int %2 %y
  %4 = load int %x
  %5 = load int %y
  %6 = sub int %4 %5
  store int %6 %3
  jump label %exit
exit:
  %7 = load int %3
  return int %7

class @Main
define void @Main.main (String[] %args)
entry:
  %foo = alloc int
  %bar = alloc int
  %arith = new @Arithmetic
  %1 = literal int 5
  store int %1 %foo
  %2 = literal int 15
  store int %2 %bar
  %3 = load int %foo
  param int %3
  %4 = load int %bar
  param int %4
  %5 = call int %arith.add
  %6 = load int %foo
  print int %6
  print String @.str.0
  %7 = load int %bar
  print int %7
  print String @.str.1
  print int %5
  print String @.str.2
  %8 = load int %foo
  param int %8
  %9 = load int %bar
  param int %9
  %10 = call int %arith.subtract
  %11 = load int %foo
  print int %11
  print String @.str.3
  %12 = load int %bar
  print int %12
  print String @.str.4
  print int %10
  jump label %exit
exit:
  return
```