

## EXAMPLE 1

```
int main() {  
    return (3+4)*5/2;  
}
```

We compile it with

```
% cc0 -b ex1.c0
```

to generate the corresponding byte code file ex1.bc0:

```
C0 C0 FF EE  # magic number  
00 13        # version 9, arch = 1 (64 bits)  
  
00 00        # int pool count  
# int pool  
  
00 00        # string pool total size  
# string pool  
  
00 01        # function count  
# function_pool  
  
#<main>  
00 00        # number of arguments = 0  
00 00        # number of local variables = 0  
00 0C        # code length = 12 bytes  
10 03        # bipush 3          # 3  
10 04        # bipush 4          # 4  
60           # iadd              # (3 + 4)  
10 05        # bipush 5          # 5  
68           # imul              # ((3 + 4) * 5)  
10 02        # bipush 2          # 2  
6C           # idiv              # (((3 + 4) * 5) / 2)  
B0           # return            #  
  
00 00        # native count  
# native pool
```

## EXAMPLE 2

```
int mid(int lo, int hi) {  
    int mid = lo + (hi - lo)/2;  
    return mid;  
}  
  
int main () {  
    return mid(3,6);  
}
```

Local variable array V = [lo, hi, mid]

Corresponding byte code for mid function  
(other parts of bytecode file not show):

```
#<mid>  
00 02    # number of arguments = 2  
00 03    # number of local variables = 3  
00 10    # code length = 16 bytes  
15 00    # vload 0      # lo  
15 01    # vload 1      # hi  
15 00    # vload 0      # lo  
64       # isub         # (hi - lo)  
10 02    # bipush 2     # 2  
6C       # idiv         # ((hi - lo) / 2)  
60       # iadd         # (lo + ((hi - lo) / 2))  
36 02    # vstore 2     # mid = ...;  
15 02    # vload 2      # mid  
B0       # return      #
```

### EXAMPLE 3

```
int next_rand(int last) {
    return last * 1664525 + 1013904223;
}

int main() {
    return next_rand(0xdeadbeef);
}
```

#### BYTECODE:

```
C0 C0 FF EE  # magic number
00 13        # version 9, arch = 1 (64 bits)

00 03        # int pool count
# int pool
00 19 66 0D
3C 6E F3 5F
DE AD BE EF

00 00        # string pool total size
# string pool

00 02        # function count
# function_pool

#<main>
00 00        # number of arguments = 0
00 01        # number of local variables = 1
00 07        # code length = 7 bytes
13 00 02 # ldc 2          # c[2] = -559038737
B8 00 01 # invokestatic 1 # next_rand(-559038737)
B0        # return      #

#<next_rand>
00 01        # number of arguments = 1
00 01        # number of local variables = 1
00 0B        # code length = 11 bytes
15 00        # vload 0    # last
13 00 00 # ldc 0          # c[0] = 1664525
68        # imul         # (last * 1664525)
13 00 01 # ldc 1          # c[1] = 1013904223
60        # iadd         # ((last * 1664525) + 1013904223)
B0        # return      #

00 00        # native count
# native pool
```

#### EXAMPLE 4

```
int main() {
    int sum = 0;
    for (int i = 1; i < 100; i += 2)
        //@loop_invariant 0 <= i && i <= 100;
        sum += i;
    return sum;
}
```

BYTECODE (only <main> shown):

```
#<main>
00 00      # number of arguments = 0
00 02      # number of local variables = 2
00 26      # code length = 38 bytes
10 00      # bipush 0          # 0
36 00      # vstore 0         # sum = 0;
10 01      # bipush 1         # 1
36 01      # vstore 1         # i = 1;
# <00:loop>
15 01      # vload 1          # i
10 64      # bipush 100       # 100
A1 00 06   # if_icmplt +6     # if (i < 100) goto <01:body>
A7 00 14   # goto +20         # goto <02:exit>
# <01:body>
15 00      # vload 0          # sum
15 01      # vload 1          # i
60         # iadd             #
36 00      # vstore 0         # sum += i;
15 01      # vload 1          # i
10 02      # bipush 2         # 2
60         # iadd             #
36 01      # vstore 1         # i += 2;
A7 FF E8   # goto -24         # goto <00:loop>
# <02:exit>
15 00      # vload 0          # sum
B0         # return           #
```

## EXAMPLE 5

```
struct point {
    int x;
    int y;
};
typedef struct point* point;

point reflect(point p) {
    point q = alloc(struct point);
    q->x = p->y;
    q->y = p->x;
    return q;
}

int main() {
    point p = alloc(struct point);
    p->x = 1;
    p->y = 2;
    point q = reflect(p);
    return q->x*10 + q->y;
}
```

BYTECODE (only <reflect> shown):

```
#<reflect>
00 01      # number of arguments = 1
00 02      # number of local variables = 2
00 1B      # code length = 27 bytes
BB 08      # new 8                # alloc(struct point)
36 01      # vstore 1             # q = alloc(struct point);
15 01      # vload 1              # q
62 00      # aaddf 0              # &q->x
15 00      # vload 0              # p
62 04      # aaddf 4              # &p->y
2E         # imload               # p->y
4E         # imstore              # q->x = p->y;
15 01      # vload 1              # q
62 04      # aaddf 4              # &q->y
15 00      # vload 0              # p
62 00      # aaddf 0              # &p->x
2E         # imload               # p->x
4E         # imstore              # q->y = p->x;
15 01      # vload 1              # q
B0         # return               #
```

## EXAMPLE 6

```
#use <conio>

int main() {
    int[] A = alloc_array(int, 100);
    for (int i = 0; i < 100; i++)
        A[i] = i;
    return A[99];
}
```

BYTECODE (only <main> shown):

```
#<main>
00 00      # number of arguments = 0
00 02      # number of local variables = 2
00 2D      # code length = 45 bytes
10 64      # bipush 100          # 100
BC 04      # newarray 4          # alloc_array(int, 100)
36 00      # vstore 0            # A = alloc_array(int, 100);
10 00      # bipush 0            # 0
36 01      # vstore 1            # i = 0;
# <00:loop>
15 01      # vload 1             # i
10 64      # bipush 100          # 100
A1 00 06   # if_icmplt +6        # if (i < 100) goto <01:body>
A7 00 15   # goto +21            # goto <02:exit>
# <01:body>
15 00      # vload 0             # A
15 01      # vload 1             # i
63         # aadds               # &A[i]
15 01      # vload 1             # i
4E         # imstore             # A[i] = i;
15 01      # vload 1             # i
10 01      # bipush 1            # 1
60         # iadd                #
36 01      # vstore 1            # i += 1;
A7 FF E7   # goto -25            # goto <00:loop>
# <02:exit>
15 00      # vload 0             # A
10 63      # bipush 99           # 99
63         # aadds               # &A[99]
2E         # imload              # A[99]
B0         # return              #
```

## EXAMPLE 7

```
#use <string>
#use <conio>

int main() {
    string h = "Hello ";
    string hw = string_join(h, "World!\n");
    print(hw);
    return string_length(hw);
}
```

### BYTECODE:

```
C0 C0 FF EE  # magic number
00 13        # version 9, arch = 1 (64 bits)

00 00        # int pool count
# int pool

00 0F        # string pool total size
# string pool
48 65 6C 6C 6F 20 00  # "Hello "
57 6F 72 6C 64 21 0A 00 # "World!\n"

00 01        # function count
# function_pool

#<main>
00 00        # number of arguments = 0
00 02        # number of local variables = 2
00 1B        # code length = 27 bytes
14 00 00     # aldc 0          # s[0] = "Hello "
36 00        # vstore 0       # h = "Hello ";
15 00        # vload 0        # h
14 00 07     # aldc 7         # s[7] = "World!\n"
B7 00 00     # invokenative 0 # string_join(h, "World!\n")
36 01        # vstore 1       # hw = ...
15 01        # vload 1        # hw
B7 00 01     # invokenative 1 # print(hw)
57          # pop             # (ignore result)
15 01        # vload 1        # hw
B7 00 02     # invokenative 2 # string_length(hw)
B0          # return          #

00 03        # native count
# native pool
00 02 00 5C  # string_join
00 01 00 10  # print
00 01 00 5D  # string_length
```