Summary

• Put a constant value into stack
  e.g. 3, -3, 0x1a
• Put a character and string into stack
  e.g. ’3’, ”hello”
• Push a value from variable
  e.g. x, a, b
• Pop a value into variable
  e.g. x=3
• Conversions
  e.g. int x=(int)1.2;
• Math operations
  e.g. x++, x%3, x>>3, x+3, x-3, x/3, x*3
Put a constant value into stack

- For small numbers
  - An integer ≤ 5
    - `iconst_m1`
    - `iconst_1`
    - `iconst_2,iconst_3,iconst_4,iconst_5`
  - A float ≤ 2
    - `fconst_0,fconst_1,fconst_2`
  - A double ≤ 1
    - `dconst_0, dconst_1`

Put a constant value into stack

- For larger numbers
  - Integer
    - `bipush <Integer number>`
    - `sipush <Integer number>`
  - Float
    - `fpush <Float number>` ????
  - Double
    - `dpush <double number>` ????
Put a constant value into stack

• For larger numbers
  – Integer
    bipush <Integer number>
sipush <Integer number>
  – Float
    fpush <Float number>
  – Double
    dpush <double number>

• ldc , ldc_w
  Put an single-word (4 Bytes) constant onto stack.
  Could be integer, float or string.
  e.g.  ldc 200000
        ldc 2.5
        ldc “hello world”

• ldc2_w
  Put a two-word (8 Bytes) constant onto stack.
  Could be long integer, double.
  e.g.  ldc2_w 2000000
        ldc2_w 2.5
Put a character into stack

• **bipush** <ASCII Code>
  
  bipush 77 ;  Ascii code of ‘M’

  Check the translated bytecode.
  Two bytes for Unicode, sipush <unicode>

• **ldc “M”**
  
  Check the translated bytecode.
  Still ASCII code of ‘M’?
  No. It is the index for constant in a constant tool.

Put a character into stack

• **Constant pool**
  
  - Constants are stored in a constant pool accessible to all methods.
  - Array-like structure.
  - Each constant has its own data type.
  - Automatically created by Jasmin.
  - Access the existing constant in pool through address(index in array of pool).
Push a value from a variable

• Load

Local variable -> stack

ilo\[\text{load}\] <index>, iload_0, iload_1, iload_2, iload_3
lload <index>, lload_0, lload_1, lload_2, lload_3
fload <index>, fload_0, fload_1, fload_2, fload_3
dload <index>, dload_0, dload_1, dload_2, dload_3
aload <index>, aload_0, aload_1, aload_2, aload_3

‘a’ means address

Pop a value to a variable

• Store

Stack -> local variable

ist\[\text{ore}\] <index>, istore_0, istore_1, istore_2, istore_3
lstore <index>, lstore_0, lstore_1, lstore_2, lstore_3
fstore <index>, fstore_0, fstore_1, fstore_2, fstore_3
dstore <index>, dstore_0, dstore_1, dstore_2, dstore_3
ast\[\text{ore}\] <index>, astore_0, astore_1, astore_2, astore_3
Conversions

• Sometimes we need to convert numbers among different types.
• **Pop** the original value from stack and **push** the converted result back onto stack.
  – long to other types
    12i, 12f, 12d
  – float to other types
    f2i, f2l, f2d
  – double to other types
    d2i, d2l, d2f
  – integer(32 bits) to other types
    i2l, i2f, i2d, i2b, i2c, i2s

Conversions

• **i2b**
  Integer to byte.
  int x = -134;
  byte b = (byte)x;

• **i2c**
  Integer to unsigned 16 bits.
  int x = -1;
  char c = (char)x;

• **i2s**
  Integer to signed 16 bits.
  int x = -40000;
  short s = (short)x;
Math operations

• Arithmetic operations
  \( X_{\text{add}}, X_{\text{sub}}, X_{\text{mul}}, X_{\text{div}}, X_{\text{rem}}, X_{\text{neg}} \) \( (X = \{i, l, f, d\}) \)
  e.g.
  
  iinc
  iinc 3 10 ; increment local variable 3 by 10

• Logical operations
  For integers and long:
  iand, ior, ixor, land, lor, lxor

• Shift operations
  ishl, ishr, lshl, lshr (Arithmetic)
  iushr, lushr (Logical)

Exercise

• Select incorrect instructions below

A. bipush 10
B. bipush 0x1a
C. bipush 0x8a
D. sipush 0x8a
E. bipush -3
F. bipush -0x1a
Exercise

• Select incorrect instructions below

A. bipush 10
   Correct.
B. bipush 0x1a
   Correct.
C. bipush 0x8a
   Wrong. Why? For a byte, the number ranges from -128 to 127. 0x8a = 8*16 + 10 = 138 > 127
D. sipush 0x8a
   Correct.
E. bipush -3
   Correct. 1111101 stored in bytecode.
F. bipush -0x1a
   Wrong. Put ‘-’ before a hex is illegal.

Exercise

• Finding errors

A.
   bipush 10
   bipush 3
   fadd

B.
   ldc 10
   bipush 3
   fadd

C.
   ldc 10
   ldc 3
   iadd

D.
   fpush 10
   bipush 3
   iadd

E.
   ldc 10
   ldc 3
   fadd

F.
   sipush 10
   sipush 3
   iadd

G.
   ldc 10.0
   ldc 3
   fadd

H.
   ldc 10.0
   ldc 3.0
   fadd
Exercise

• Finding errors

A. Wrong
bipush 10
bipush 3
fadd
iadd

B. Wrong
ldc 10
bipush 3
fadd
iadd

C. ldc 10
ldc 3
iadd

D. Wrong
fpush
ldc 10
bipush 3
iadd

E. Wrong
ldc 10.0
ldc 3.0
fadd

F. sipush 10
sipush 3
iadd

G. Wrong
ldc 10.0
ldc 3.0
fadd

H. ldc 10.0
ldc 3.0
fadd

Exercise

• Finding errors

; int x=(int)1.0
fconst_1 ; Put a constant 1.0 into stack
istore_1 ; Assign local variable 1 as 1.0
Exercise

• Finding errors

```java
; int x=(int)1.0
  fconst_1 ; Put a constant 1.0 into stack
  istore_1 ; Assign local variable 1 as 1.0
```

Wrong.
fconst_1 is for a float number.
istore_1 is for an integer number.
But they are accessing the same number.
So conversion is needed.

```java
; int x=(int)1.0
  fconst_1 ; Put a constant 1.0 into stack
  f2i ; Convert a float to integer type
  istore_1 ; Assign local variable 1 as 1.0
```