**Question 3**

1. Explanation:

BIPUSH 0

ISTORE i

*//up to this line, i now store the value 0*

BIPUSH 0

ISTORE j

*//up to this line, j now store the value 0*

L1: BIPUSH 6

ILOAD j

ISUB

*//calculate the value 6-j*

IFLT L2

*//if 6-j is less than zero, then go to L2. Or else, if 6-j is greater than or equal to 0, execute the code below*

ILOAD i

BIPUSH 3

IADD

*//calculate the value i+3*

DUP

*//calculate i+3 one more time*

IADD

*//add i+3 to i+3*

ISTORE i

*//store that back in i*

IINC j 2

*//increment j by 2*

GOTO L1

*//this will make the program keep going back to L1 until the IFLT check is True, then it goes to L2. So this sort of make it a loop*

L2:

…

The code in Python:

i = 0

j = 0

while 6 >= j

i = (i + 3) + (i + 3)

j = j + 2

else

…..

1. Follow the table in the handout

|  |  |  |
| --- | --- | --- |
| **Address**  **in Method Area (hex)** | **Opcode stored in method area**  **(hex)** | **Mnemonic** |
| 100 | 10 00 | BIPUSH 0 |
| 102 | 36 00 | ISTORE i |
| 104 | 10 00 | BIPUSH 0 |
| 106 | 36 01 | ISTORE j |
| 108 | 10 06 | L1: BIPUSH 6 |
| 10A | 15 01 | ILOAD j |
| 10C | 64 | ISUB |
| 10D | 9B | IFLT L2 |
| 110 | 15 00 | ILOAD i |
| 112 | 10 03 | BIPUSH 3 |
| 114 | 60 | IADD |
| 115 | 59 | DUP |
| 116 | 60 | IADD |
| 117 | 36 00 | ISTORE i |
| 119 | 84 01 02 | IINC j 2 |
| 11C | A7 FF EE | GOTO L1 |
| 11F |  | L2 |

* Address calculation for GOTO L1:

+ Go back to L1 which is declared above → adjust 18 locations backward

→ Moving PC to -18 locations

+ +18 in binary: 0000 0000 0001 0010

→ two’s complement: 1111 1111 1110 1110 = FFEE

* Address calculation for IFLT L2

+ Advance to L2 which is declared below

11F

- 10D

012

1. Tracing

|  |  |  |
| --- | --- | --- |
| Address of instruction leading to change (hex) | i (hex) | j (hex) |
| 100 | - | - |
| 102 | 0 | - |
| 106 | 0 | 0 |
| 117 | 6 | 0 |
| 119 | 6 | 2 |
| 117 | 18 | 2 |
| 119 | 18 | 4 |
| 117 | 42 | 4 |
| 119 | 42 | 6 |
| 117 | 90 | 6 |
| 119 | 90 | 8 |