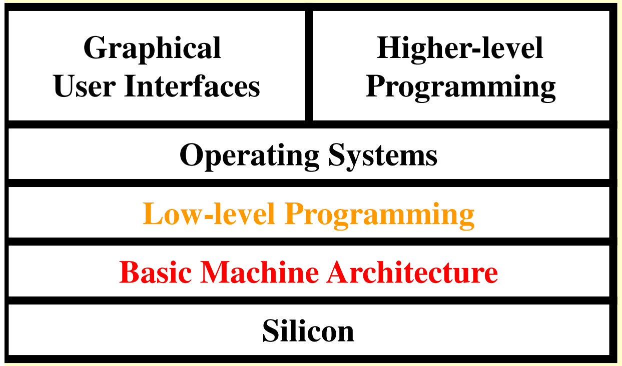


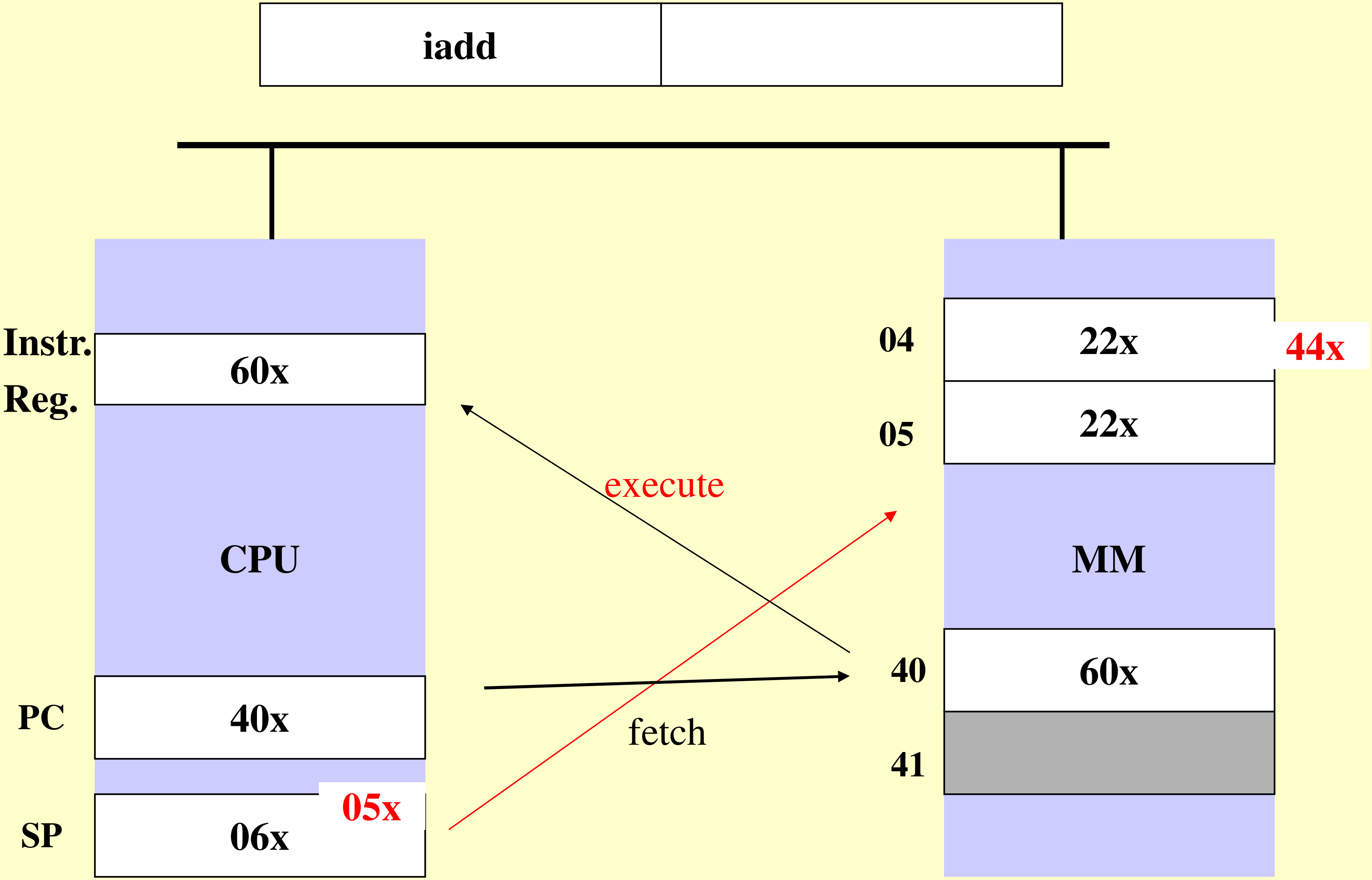
CSCU9V4
Systems

Systems lecture 11

Computer Organisation

A Virtual Machine

a demonstration of V4 Virtual machine


iadd
Instr.
Reg.
60x
CPU
PC
40x
SP
06x
05x
execute
fetch
04
22x
44x
05
22x
MM
40
60x
41

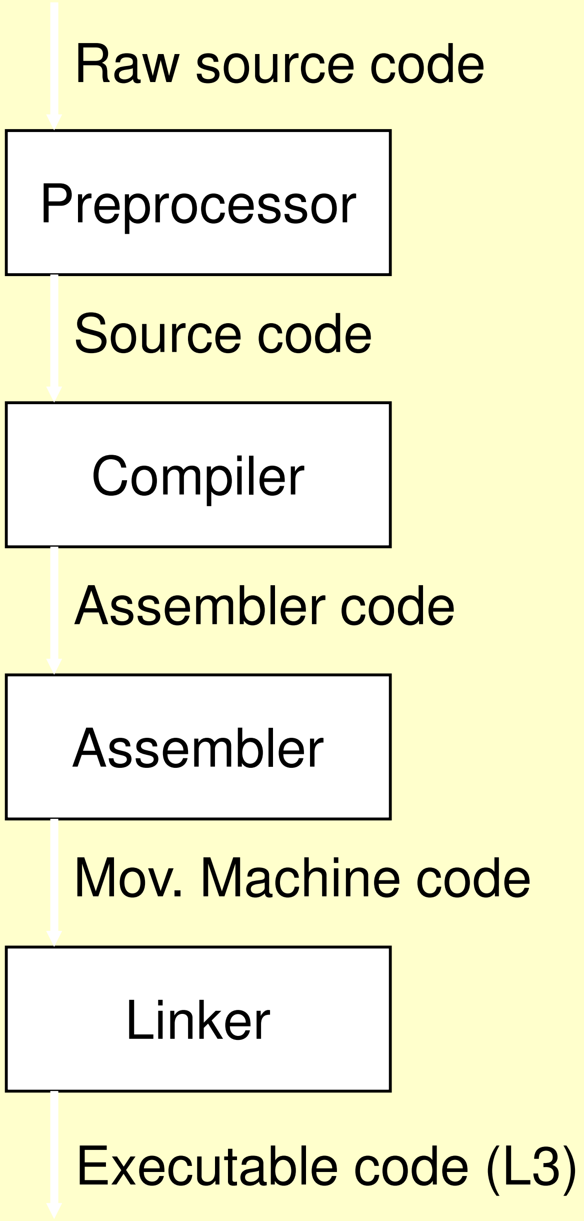
Most of the instructions (some more later)

Instruction (in hex)	Mnemonic	Translation
60ii	iadd	addition of top two stack elements (pops two numbers from the top of the stack, and pushes their sum back onto the stack)
64ii	isub	subtraction of top two stack elements
68ii	imul	multiplication of top two stack elements
6Cii	idiv	division of top two stack elements
70ii	irem	remainder operation on top two stack elements
7Eii	iand	boolean AND operation on top two stack elements
80ii	ior	boolean OR operation on top two stack elements
12xx	lcd #	pushes constant value xx onto top of stack
15xx	iload #	pushes a copy of memory location (localVariablesPointer+xx) onto top of stack
36xx	istore #	pops the top element of the stack and stores it in memory location (localVariablesPointer+xx)
a7xx	goto #	sets program counter to memory address xx
99xx	ifeq #	pops the top element of the stack, and if it equals zero, sets the program counter to xx

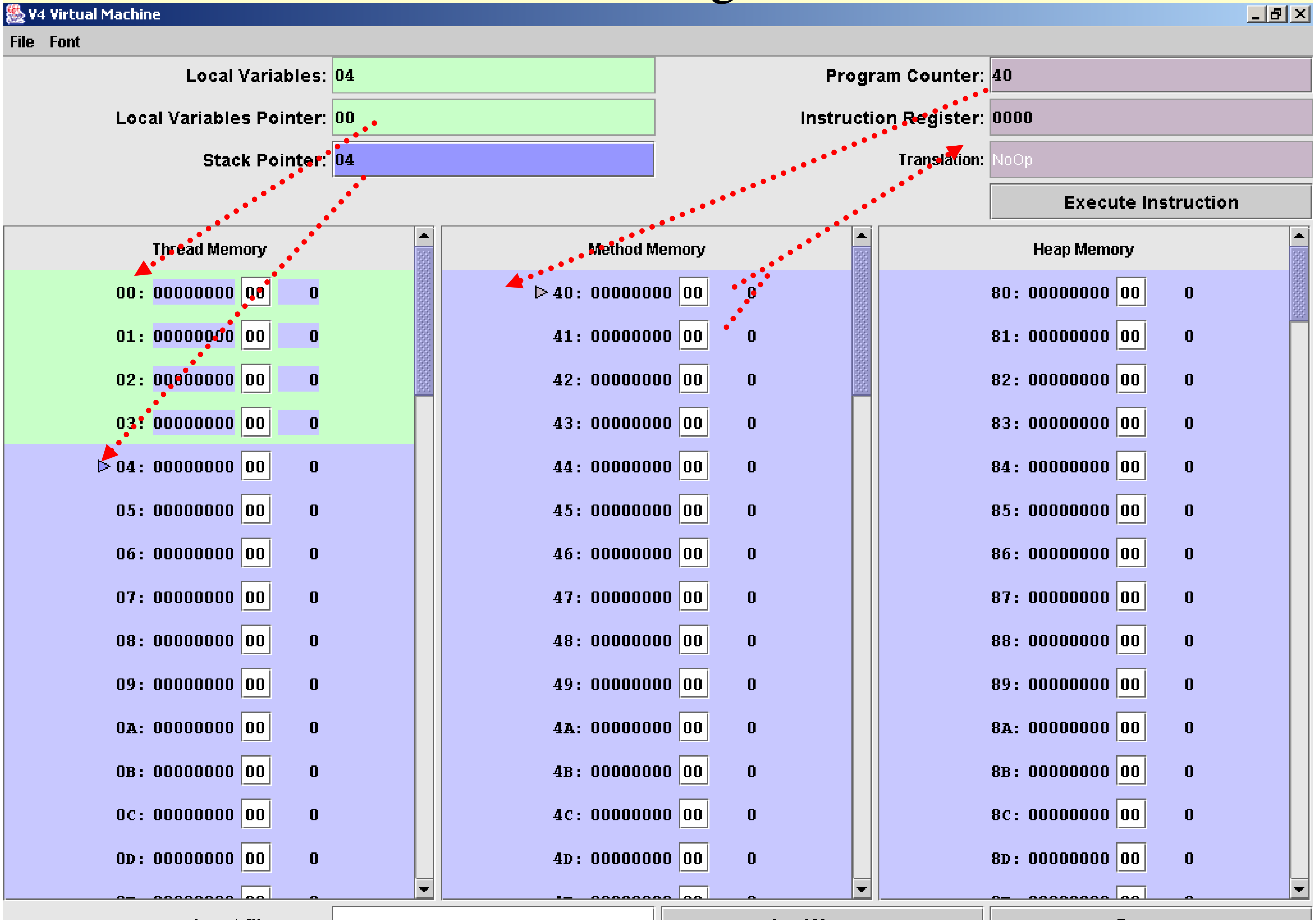
- ii: any two hex digits (ignored)
- xx: two hex digits used either as a constant (lcd) or address

An Example

- High-level language:
  - some_var = 15 + 10
- Assembly language
  - ldc F
  - ldc A
  - iadd
  - istore local_var+0
- Movable machine code
  - 0001 0010 00001111 * (push Fx onto stack)
  - 0001 0010 00001010 * (push Ax onto stack)
  - 0110 0000 00000000 * (add top 2 stack element)
  - 0011 0110 00000000 * (store at first local var)
- Executable machine code (start at L=00001111)
  - 40x 0001 0010 00001111 * (push Ex onto stack)
  - 42x 0001 0010 00001010 * (push Ax onto stack)
  - 44x 0110 0000 00000000 * (add top 2 stack element)
  - 46x 0011 0110 00000000 * (store at first local var)



the resulting GUI



Local Variables: 04

Local Variables Pointer: 00

Stack Pointer: 04

Program Counter: 40

Instruction Register: 0000

Translation: NoOp

Thread Memory

00:	00000000	00	0
01:	00000000	00	0
02:	00000000	00	0
03:	00000000	00	0
▷ 04:	00000000	00	0
05:	00000000	00	0
06:	00000000	00	0
07:	00000000	00	0
08:	00000000	00	0
09:	00000000	00	0
0A:	00000000	00	0
0B:	00000000	00	0
0C:	00000000	00	0
0D:	00000000	00	0

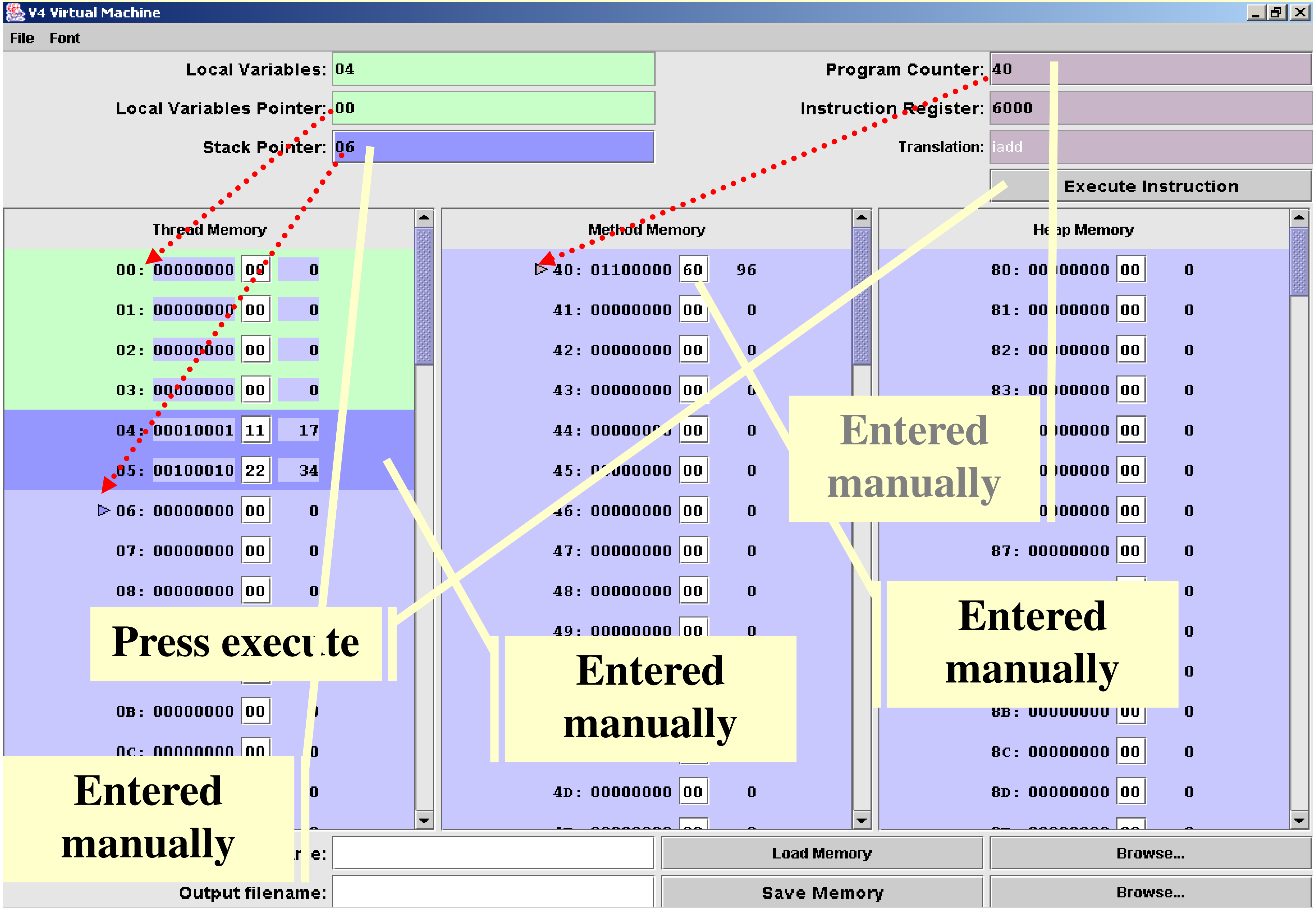
Method Memory

▷ 40:	00000000	00	0
41:	00000000	00	0
42:	00000000	00	0
43:	00000000	00	0
44:	00000000	00	0
45:	00000000	00	0
46:	00000000	00	0
47:	00000000	00	0
48:	00000000	00	0
49:	00000000	00	0
4A:	00000000	00	0
4B:	00000000	00	0
4C:	00000000	00	0
4D:	00000000	00	0

Heap Memory

80:	00000000	00	0
81:	00000000	00	0
82:	00000000	00	0
83:	00000000	00	0
84:	00000000	00	0
85:	00000000	00	0
86:	00000000	00	0
87:	00000000	00	0
88:	00000000	00	0
89:	00000000	00	0
8A:	00000000	00	0
8B:	00000000	00	0
8C:	00000000	00	0
8D:	00000000	00	0

before we execute iadd



Local Variables: 04

Local Variables Pointer: 00

Stack Pointer: 06

Program Counter: 40

Instruction Register: 6000

Translation: iadd

Thread Memory

```
   00: 00000000 00   0
   01: 00000000 00   0
   02: 00000000 00   0
   03: 00000000 00   0
   04: 00010001 11  17
   05: 00100010 22  34
▷ 06: 00000000 00   0
   07: 00000000 00   0
   08: 00000000 00   0
   0B: 00000000 00
   0C: 00000000 00   0
```

Method Memory

```
▷ 40: 01100000 60  96
   41: 00000000 00   0
   42: 00000000 00   0
   43: 00000000 00   0
   44: 00000000 00   0
   45: 00000000 00   0
   46: 00000000 00   0
   47: 00000000 00   0
   48: 00000000 00   0
   49: 00000000 00   0
   4D: 00000000 00   0
```

Heap Memory

```
80: 00000000 00   0
81: 00000000 00   0
82: 00000000 00   0
83: 00000000 00   0
    00000000 00   0
    00000000 00   0
    00000000 00   0
87: 00000000 00   0
                  0
                  0
                  0
8B: 00000000 00   0
8C: 00000000 00   0
8D: 00000000 00   0
```

Input filename:

Output filename:

Entered manually

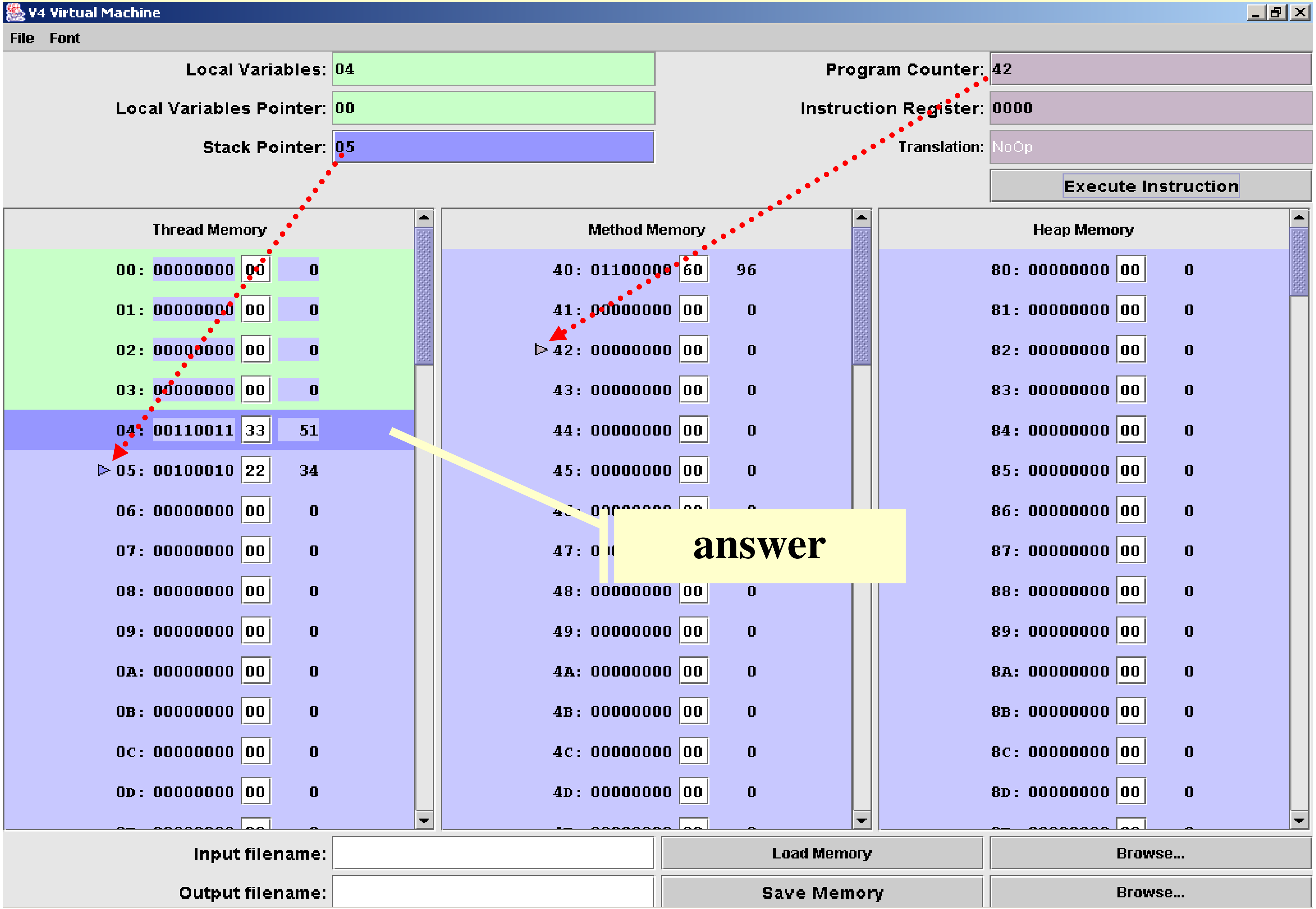
Entered manually

Entered manually

Entered manually

Press execute

after we execute iadd



Local Variables: 04

Local Variables Pointer: 00

Stack Pointer: 05

Program Counter: 42

Instruction Register: 0000

Translation: NoOp

Thread Memory

00:	00000000	00	0
01:	00000000	00	0
02:	00000000	00	0
03:	00000000	00	0
04:	00110011	33	51
▷ 05:	00100010	22	34
06:	00000000	00	0
07:	00000000	00	0
08:	00000000	00	0
09:	00000000	00	0
0A:	00000000	00	0
0B:	00000000	00	0
0C:	00000000	00	0
0D:	00000000	00	0

Method Memory

40:	01100000	60	96
41:	00000000	00	0
▷ 42:	00000000	00	0
43:	00000000	00	0
44:	00000000	00	0
45:	00000000	00	0
[illegible]	[illegible]	[illegible]	0
47:	[illegible]	[illegible]	[illegible]
48:	00000000	00	0
49:	00000000	00	0
4A:	00000000	00	0
4B:	00000000	00	0
4C:	00000000	00	0
4D:	00000000	00	0

Heap Memory

80:	00000000	00	0
81:	00000000	00	0
82:	00000000	00	0
83:	00000000	00	0
84:	00000000	00	0
85:	00000000	00	0
86:	00000000	00	0
87:	00000000	00	0
88:	00000000	00	0
89:	00000000	00	0
8A:	00000000	00	0
8B:	00000000	00	0
8C:	00000000	00	0
8D:	00000000	00	0

Input filename:

Output filename:

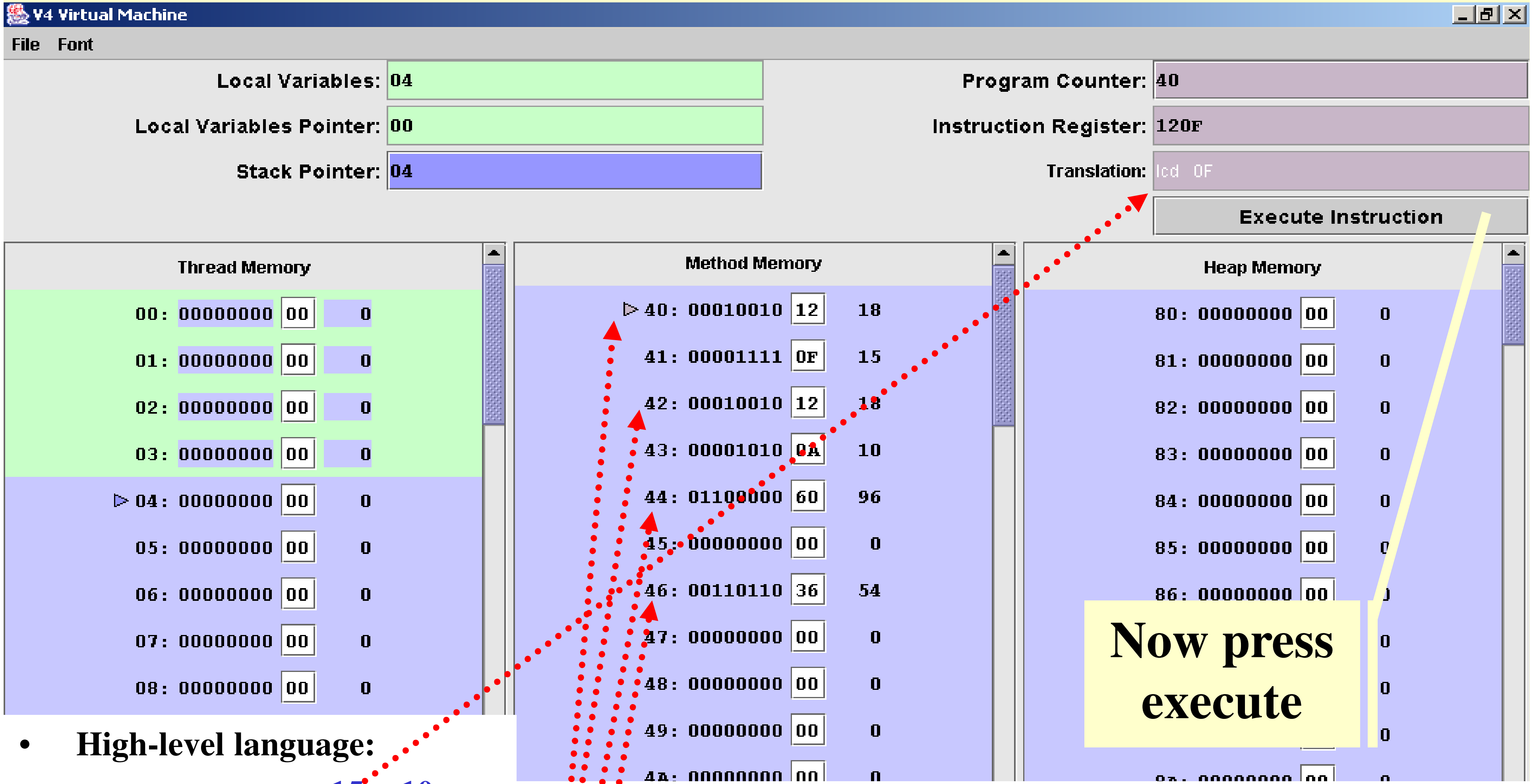
answer

… but we put the data manually on the stack!

- **High-level language:**
  - **some_var := 15 + 10**
- **Assembly language**
  - **ldc F**
  - **ldc A**
  - **iadd**
  - **istore local_var+0**
- **Movable machine code**
  - **0001 0010 00001111 * (push Ex onto stack)**
  - **0001 0010 00001010 * (push Ax onto stack)**
  - **0110 0000 00000000 * (add top 2 stack element)**
  - **0011 0110 00000000 * (store at first local var)**
- **Executable machine code (start at L=00001111)**
  - **40x 0001 0010 00001111 * (push Ex onto stack)**
  - **42x 0001 0010 00001010 * (push Ax onto stack)**
  - **44x 0110 0000 00000000 * (add top 2 stack element)**
  - **46x 0011 0110 00000000 * (store at first local var)**

… here the program puts
the data on the stack.

At start …..



V4 Virtual Machine

File Font

Local Variables: 04

Local Variables Pointer: 00

Stack Pointer: 04

Program Counter: 40

Instruction Register: 120F

Translation: lcd 0F

Execute Instruction

Thread Memory

```
   00: 00000000 00  0
   01: 00000000 00  0
   02: 00000000 00  0
   03: 00000000 00  0
▷ 04: 00000000 00  0
   05: 00000000 00  0
   06: 00000000 00  0
   07: 00000000 00  0
   08: 00000000 00  0
```

Method Memory

```
▷ 40: 00010010 12  18
   41: 00001111 0F  15
   42: 00010010 12  18
   43: 00001010 0A  10
   44: 01100000 60  96
   45: 00000000 00   0
   46: 00110110 36  54
   47: 00000000 00   0
   48: 00000000 00   0
   49: 00000000 00   0
   4A: 00000000 00   0
```

Heap Memory

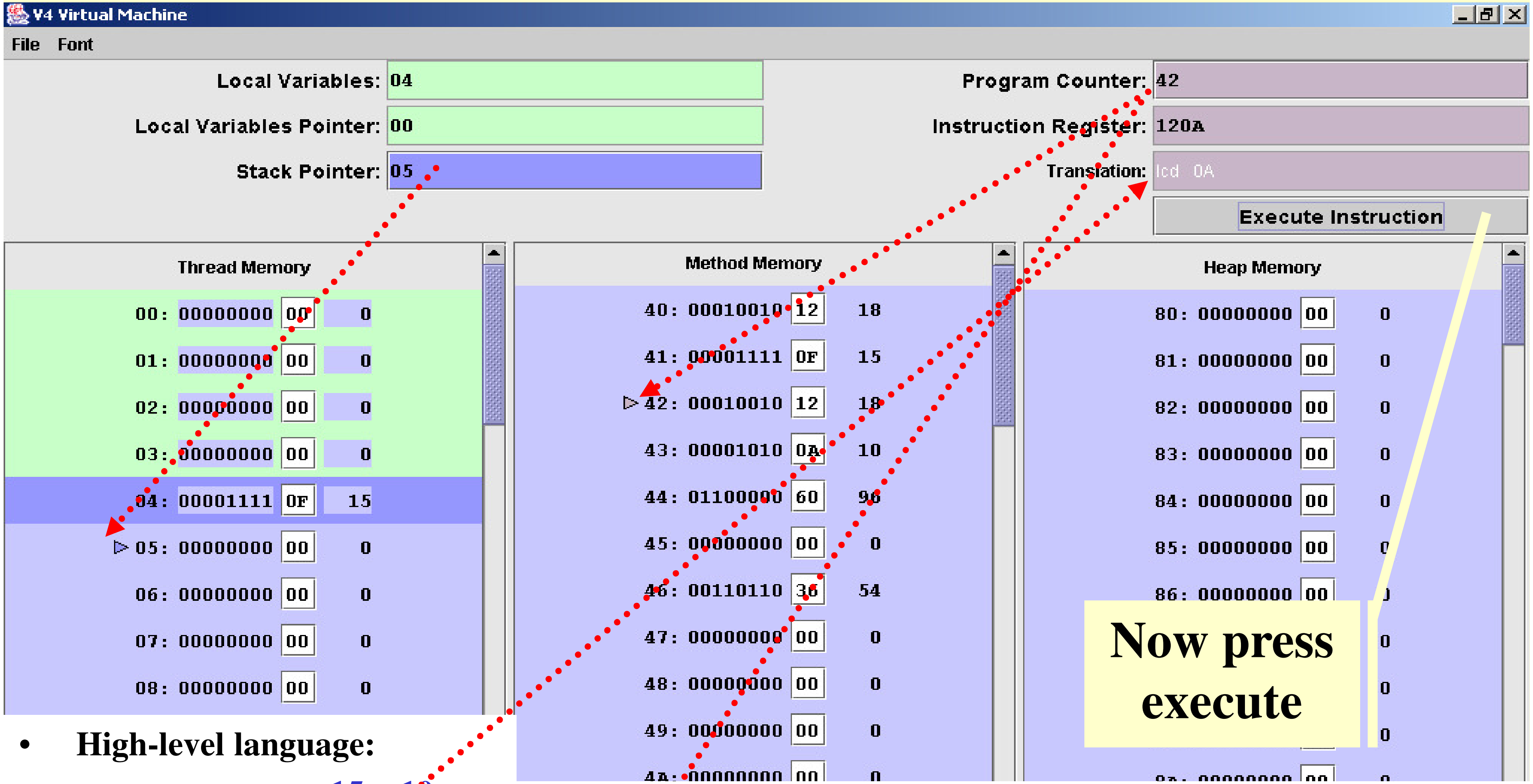
```
80: 00000000 00  0
81: 00000000 00  0
82: 00000000 00  0
83: 00000000 00  0
84: 00000000 00  0
85: 00000000 00  0
86: 00000000 00  0
                 0
                 0
                 0
8A: 00000000 00  0
```

Now press execute

- High-level language:
  - some_var := 15 + 10
- Assembly language
  - ldc F
  - ldc A
  - iadd
  - istore local_var+0

- Executable machine code (start at L=00001111)
  - 40x 0001 0010 00001111 * (push Fx onto stack)
  - 42x 0001 0010 00001010 * (push Ax onto stack)
  - 44x 0110 0000 00000000 * (add top 2 stack element)
  - 46x 0011 0110 00000000 * (store at first local var)

Executed 1st instruction …..



Local Variables: 04

Local Variables Pointer: 00

Stack Pointer: 05

Program Counter: 42

Instruction Register: 120A

Translation: lcd 0A

Thread Memory

00:	00000000	0F	0
01:	00000000	00	0
02:	00000000	00	0
03:	00000000	00	0
04:	00001111	0F	15
▷ 05:	00000000	00	0
06:	00000000	00	0
07:	00000000	00	0
08:	00000000	00	0

Method Memory

40:	00010010	12	18
41:	00001111	0F	15
▷ 42:	00010010	12	18
43:	00001010	0A	10
44:	01100000	60	96
45:	00000000	00	0
46:	00110110	36	54
47:	00000000	00	0
48:	00000000	00	0
49:	00000000	00	0
4A:	00000000	00	0

Heap Memory

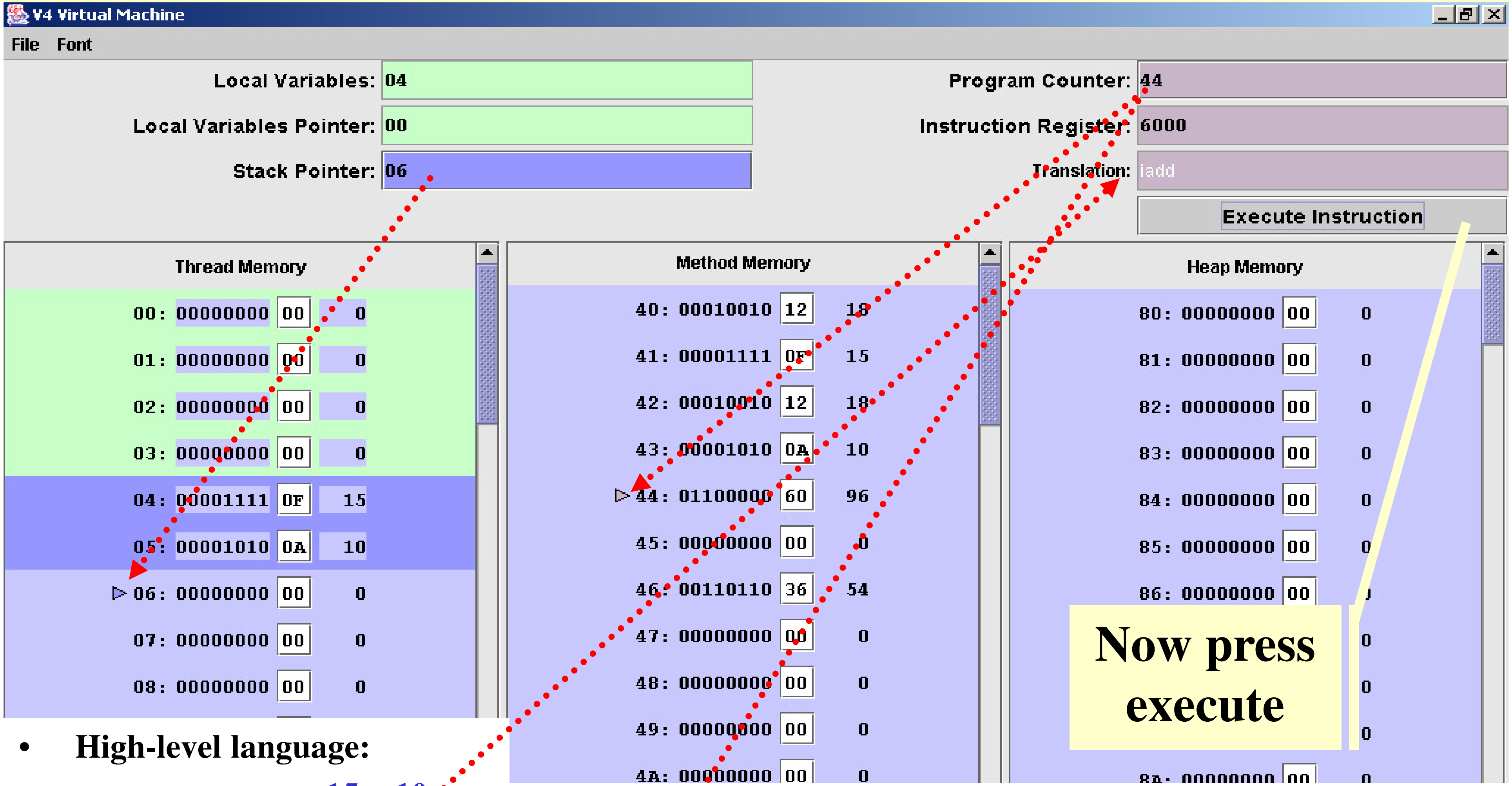
80:	00000000	00	0
81:	00000000	00	0
82:	00000000	00	0
83:	00000000	00	0
84:	00000000	00	0
85:	00000000	00	0
86:	00000000	00	0

Now press execute

- **High-level language:**
  - some_var := 15 + 10
- **Assembly language**
  - ldc F
  - ldc A
  - iadd
  - istore local_var+0

- **Executable machine code (start at L=00001111)**
  - 40x 0001 0010 00001111 * **(push Fx onto stack)**
  - **42x 0001 0010 00001010 * (push Ax onto stack)**
  - **44x 0110 0000 00000000 * (add top 2 stack element)**
  - **46x 0011 0110 00000000 * (store at first local var)**

Executed 2nd instruction …..



V4 Virtual Machine

File Font

Local Variables: 04

Local Variables Pointer: 00

Stack Pointer: 06

Program Counter: 44

Instruction Register: 6000

Translation: iadd

Thread Memory

00:	00000000	00	0
01:	00000000	00	0
02:	00000000	00	0
03:	00000000	00	0
04:	00001111	0F	15
05:	00001010	0A	10
▷ 06:	00000000	00	0
07:	00000000	00	0
08:	00000000	00	0

Method Memory

40:	00010010	12	18
41:	00001111	0F	15
42:	00010010	12	18
43:	00001010	0A	10
▷ 44:	01100000	60	96
45:	00000000	00	0
46:	00110110	36	54
47:	00000000	00	0
48:	00000000	00	0
49:	00000000	00	0
4A:	00000000	00	0

Heap Memory

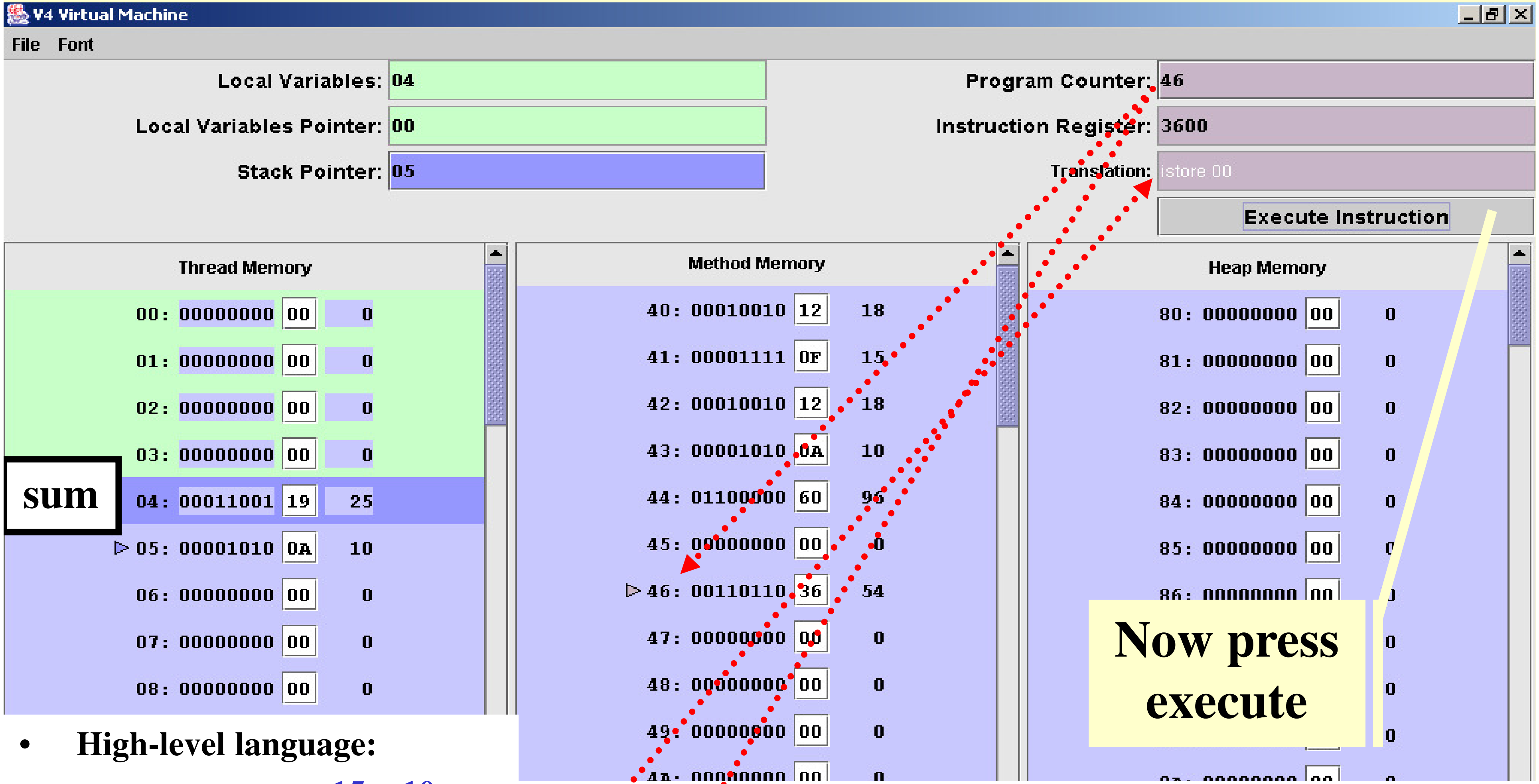
80:	00000000	00	0
81:	00000000	00	0
82:	00000000	00	0
83:	00000000	00	0
84:	00000000	00	0
85:	00000000	00	0
86:	00000000	00	0
			0
			0
			0
8A:	00000000	00	0

Now press execute

- High-level language:
  - some_var := 15 + 10
- Assembly language
  - ldc F
  - ldc A
  - iadd
  - istore local_var+0

- Executable machine code (start at L=00001111)
  - 40x 0001 0010 00001111 * (push Fx onto stack)
  - 42x 0001 0010 00001010 * (push Ax onto stack)
  - 44x 0110 0000 00000000 * (add top 2 stack element)
  - 46x 0011 0110 00000000 * (store at first local var)

Executed 3rd instruction …..



Local Variables: 04

Local Variables Pointer: 00

Stack Pointer: 05

Program Counter: 46

Instruction Register: 3600

Translation: istore 00

Thread Memory

00:	00000000	00	0
01:	00000000	00	0
02:	00000000	00	0
03:	00000000	00	0
04:	00011001	19	25
▷ 05:	00001010	0A	10
06:	00000000	00	0
07:	00000000	00	0
08:	00000000	00	0

Method Memory

40:	00010010	12	18
41:	00001111	0F	15
42:	00010010	12	18
43:	00001010	0A	10
44:	01100000	60	96
45:	00000000	00	0
▷ 46:	00110110	36	54
47:	00000000	00	0
48:	00000000	00	0
49:	00000000	00	0
4A:	00000000	00	0

Heap Memory

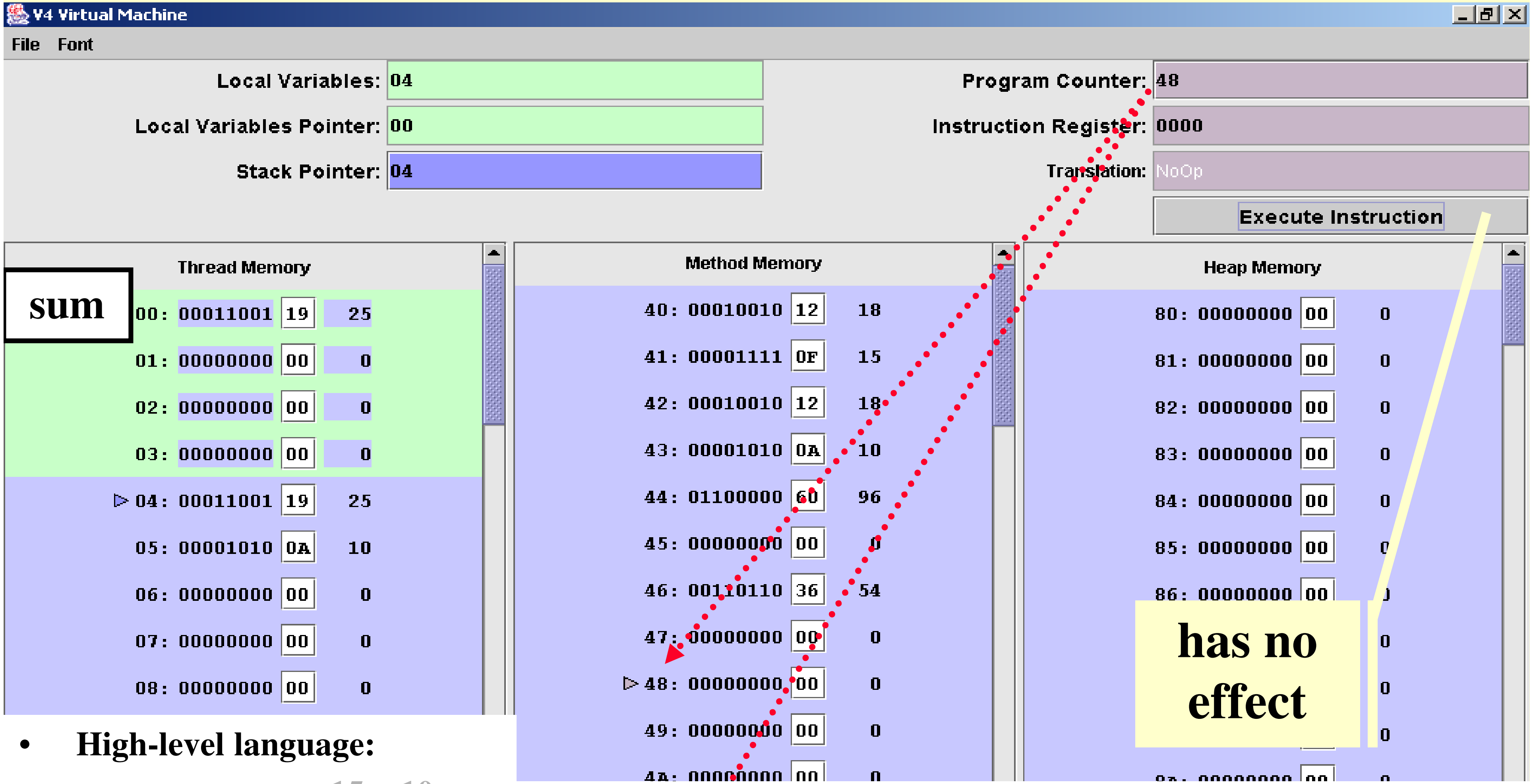
80:	00000000	00	0
81:	00000000	00	0
82:	00000000	00	0
83:	00000000	00	0
84:	00000000	00	0
85:	00000000	00	0
86:	00000000	00	0

sum

Now press execute

- High-level language:
  - some_var := 15 + 10
- Assembly language
  - ldc F
  - ldc A
  - iadd
  - istore local_var+0
- Executable machine code (start at L=00001111)
  - 40x 0001 0010 00001111 * (push Fx onto stack)
  - 42x 0001 0010 00001010 * (push Ax onto stack)
  - 44x 0110 0000 00000000 * (add top 2 stack element)
  - 46x 0011 0110 00000000 * (store at first local var)

Executed last instruction …..



V4 Virtual Machine

File Font

Local Variables: 04

Local Variables Pointer: 00

Stack Pointer: 04

Program Counter: 48

Instruction Register: 0000

Translation: NoOp

Execute Instruction

Thread Memory

sum

```
   00: 00011001 19  25
   01: 00000000 00   0
   02: 00000000 00   0
   03: 00000000 00   0
▷ 04: 00011001 19  25
   05: 00001010 0A  10
   06: 00000000 00   0
   07: 00000000 00   0
   08: 00000000 00   0
```

Method Memory

```
   40: 00010010 12  18
   41: 00001111 0F  15
   42: 00010010 12  18
   43: 00001010 0A  10
   44: 01100000 60  96
   45: 00000000 00   0
   46: 00110110 36  54
   47: 00000000 00   0
▷ 48: 00000000 00   0
   49: 00000000 00   0
   4A: 00000000 00   0
```

Heap Memory

```
80: 00000000 00   0
81: 00000000 00   0
82: 00000000 00   0
83: 00000000 00   0
84: 00000000 00   0
85: 00000000 00   0
86: 00000000 00   0
                  0
                  0
                  0
8A: 00000000 00   0
```

has no effect

- High-level language:
  - some_var := 15 + 10
- Assembly language
  - ldc F
  - ldc A
  - iadd
  - istore local_var+0

- Executable machine code (start at L=00001111)
  - 40x 0001 0010 00001111 * (push Fx onto stack)
  - 42x 0001 0010 00001010 * (push Ax onto stack)
  - 44x 0110 0000 00000000 * (add top 2 stack element)
  - 46x 0011 0110 00000000 * (store at first local var)

Example program

```
40
1234 ldc    34 place constant (34x) on stack
12df ldc    df place constant (dfX) on stack
6000 iadd add top two stack values
3602 istore 02 store value at location 2
```

- The first two hex digits define where the program will be loaded
- The remainder is the program
- Note that only the hex digits at the start of the line are used: the rest of the line is ignored.

Consider a program fragment …

- Say we have a fragment of code

```
X = 2
if X != 0
  y = 4 ;
else y = 7 ;
```

How does this get translated into low-level language?

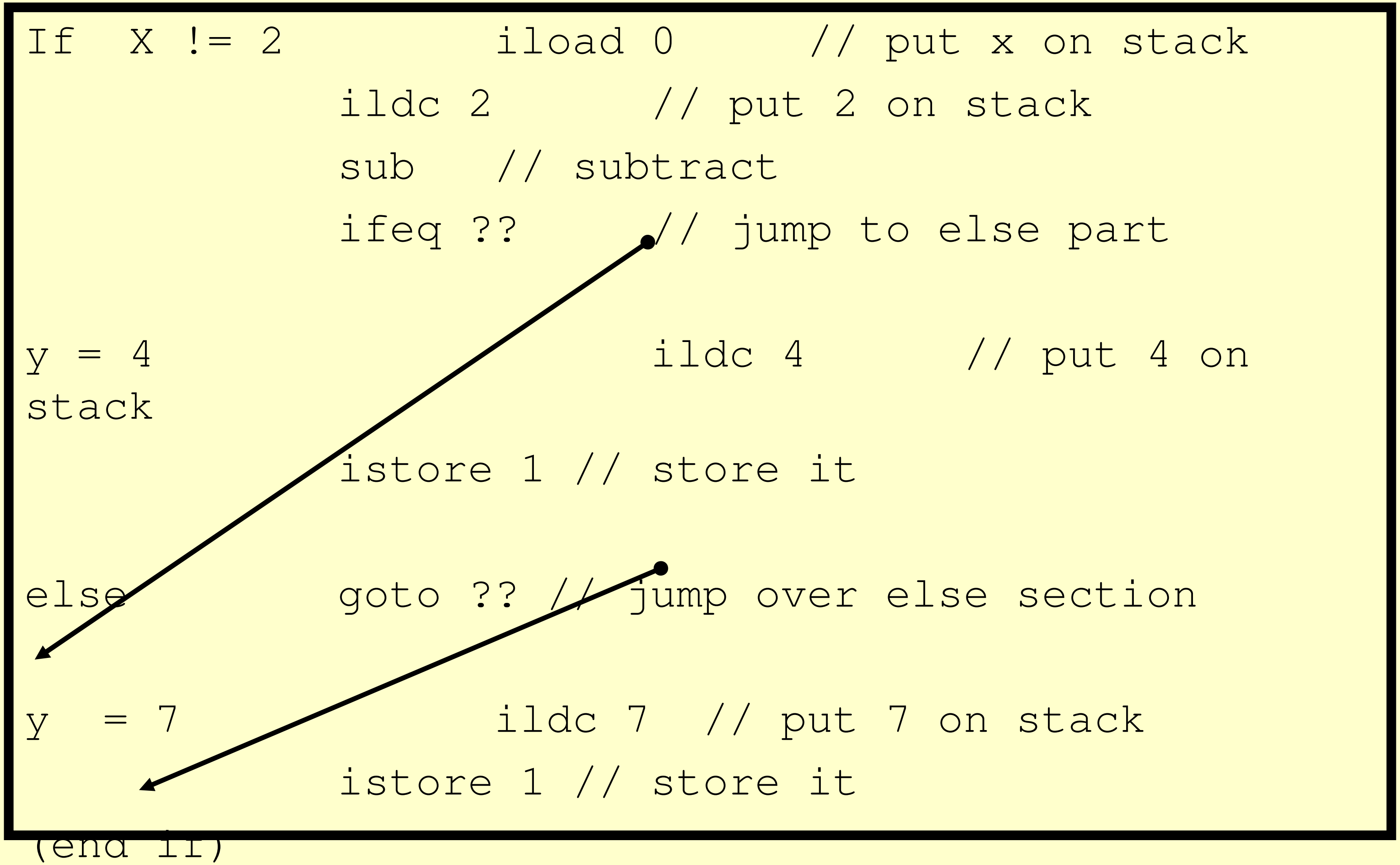
Translating

- What needs to be done?
    - assign locations to variables

      `X, y:` Put X at location 0, and y at location 1
    - Translate line by line.

```
X = 2        ildc 2  // push 2 on to stack
             istore 0  // store in location 0
```

Translating continued



```
If   X != 2          iload 0      // put x on stack
            ildc 2       // put 2 on stack
            sub   // subtract
            ifeq ??      // jump to else part

y = 4                      ildc 4       // put 4 on
stack
            istore 1 // store it

else        goto ?? // jump over else section

y  = 7               ildc 7  // put 7 on stack
            istore 1 // store it
(end if)
```

And into machine code

```
40          // start address for loading
1202    // ildc 2 // push 2 on to stack
3600    //istore 0      // store in location 0
   1502 //iload 0 // put x on stack
   1202 //ildc 2  // put 2 on stack
   6400 //isub          // subtract
   99xx       // ifeq ??      // jump to else
part
```

- We don’t know where the jump is to until we generates some more code.

Machine code cont’d

```
1204  //ildc 4          // put 4 on stack
3601        //istore 1  // store it

A7xx  //Goto xx jump to end of if statement
```

- Again, we don’t know where to exactly

```
 1207  //ildc 7          // put 7 on
 stack
 3601  //istore 1  // store it

 0000  // do nothing
```

Resolving

```
        40               // start address for loading
40      1202     // ildc 2   // push 2 on to stack
42      3600     //istore 0          // store in location 0
44         1502  //iload 2   // put x on stack
46         1202  //ildc 2    // put 2 on stack
48         6400  //isub                 // subtract
4A         9952       // ifeq 0x52     // jump to
4C      else part
4E         1204  //ildc 4    // put 4 on stack
50         3601       //istore 1       // store it
52         A756  //Goto 56   // jump to end of if
54      statement
56         1207  //ildc 7    // put 7 on stack
           3601  //istore 1          // store it
           0000       // do nothing
```

End of lecture