**Machine Code Fundamentals**

For each of the code snippets on the left, fill in what the machine code would look like in the memory on the right. Use the `Instruction Set Reference to get the syntax right and **include the name of variables** in parentheses next to their address number.

|  |  |
| --- | --- |
| **Address** | **Instructions/Data** |
| 0 | store 9 mem[7] |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 |  |
| 7 (y) |  |
| 8 (x) |  |

x = 9

y = 6

y = y \* x

x = x \* x

print(x)

print(y)

|  |  |
| --- | --- |
| **Address** | **Instructions/Data** |
| 0 |  |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 |  |
| 7 |  |
| 8 |  |

x = 0

y = 1

while x != 3:

y = y \* 2

x = x + 1

print(x)

|  |  |
| --- | --- |
| **Address** | **Instructions/Data** |
| 0 |  |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 |  |
| 7 |  |
| 8 |  |
| 9 |  |
| 10 |  |
| 11 |  |

l1 = [1, 2, 3]

l2 = l1

l2[3] = 100

print(l1[3])

print(l2[3])

|  |  |
| --- | --- |
| **Address** | **Instructions/Data** |
| 0 |  |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 |  |
| 7 |  |
| 8 |  |
| 9 (pete) |  |
| 10 |  |
| 11 |  |

class Pet(typing.NamedTuple):

age: int

num\_legs: int

pete = Pet(5, 4)

if pete.age == 7:

print(1000)

else:

print(-1)

|  |  |
| --- | --- |
| **Address** | **Instructions/Data** |
| 0 |  |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 |  |
| 7 |  |
| 8 |  |
| 9 |  |
| 10 |  |
| 11 |  |
| 12 |  |
| 13 |  |

class Pet(typing.NamedTuple):

age: int

num\_legs: int

pets = [Pet(5, 4), Pet(1, 100)]

print(pets[0].num\_legs)

print(pets[1].age)

**Instruction Set Reference**

* **Unconditional jumps**
  + Format
    - jmp 🡪 PC-\_
  + Example
    - jmp 🡪 PC-8
  + Instructions
    - jmp
* **Conditional jumps** 
  + Format:
    - instruction VALUE , VALUE 🡪 PC-\_\_
  + Example
    - jneq mem[8], mem[9] 🡪 PC-8
  + Instructions
    - jneq, jeq, jle, jlt, jge, jgt
* **Memory Store**
  + Format
    - instruction VALUE 🡪 mem[\_]
  + Example
    - store mem[8] 🡪 mem[9]
    - store mem[mem[10] + 1] 🡪 mem[mem[10] + 2]
  + Instructions
    - store
* **Operations**
  + Format
    - instruction VALUE, VALUE 🡪 mem[\_]
  + Example
    - add mem[1], 10 🡪 mem[1]
  + Instructions
    - add
    - sub
    - mul
    - div
* **Output**
  + Format
    - syscall operation VALUE
  + Example
    - syscall print mem[1]
    - syscall print mem[mem[10] + 5]
  + Instructions
    - print
    - terminate