CPU Dojo 4

We're now going to build an assembler for our CPU. This will allow us to write more complicated programs.

1. Code from dojo 1
Let's start with a simple example from dojo 1. Parse and compile the following assembly into our CPU's machine code.

```
LDA 100
ADC 7
STA 5
BRK
```

Expected output: `1, 100, 2, 7, 3, 5, 0`

2. Code from dojo 2
Let's take it up a notch and use the code from dojo 2. Below is a snippet of the code you can download the full code here: http://bit.ly/18CH5TS

```
...
LDA 32
STA_X
INX
DEY
CMY 0
BNE -21
BRK
...
```

3. Use labels as references for branching
In the previous example the assembly is using the number -21 as a memory address offset for the BNE opcode. In real assembly you would use an alphanumeric label as the value for the BNE opcode and the assembler would calculate the necessary memory address offset for you.

Let's write a program which can compile the following assembly

```
LDY 3
loop:
  DEY
  CMY 0
  BNE loop
BRK
```

Expected output: `10, 3, 9, 6, 0, 7, -5, 0`

4. Use labels as references for subroutines

4.1 Single subroutine reference

```
LDA 10
JSR incABy10
BRK

incABy10:
  ADC 10
  RTS
```

Expected output: `1, 10, 11, 5, 0, 2, 10, 12`

4.2 Nested subroutine references

```
LDA 10
JSR incABy10
BRK

incABy10:
  ADC 10
  JSR incABy50
  RTS

incABy50:
  ADC 50
  RTS
```

Expected output: `1, 10, 11, 5, 0, 2, 10, 11, 10, 12, 2, 50, 12`

4.3 Who let the dogs out, assembly style

I've converted the code from Dojo 2 into assembly for you to run.
You can find the code here: http://bit.ly/1L0HLV4

Expected output:

```
4, 128, 11, 32, 11, 101, 11, 45, 11, 101, 11, 58, 11, 101, 11, 71,
11, 101, 11, 88, 10, 3, 9, 11, 101, 11, 32, 6, 0, 7, -9, 0, 1,
119, 11, 106, 1, 104, 11, 106, 1, 111, 11, 106, 12, 1, 108, 11,
106, 1, 101, 11, 106, 1, 116, 11, 106, 12, 1, 116, 11, 106, 1,
104, 11, 106, 1, 101, 11, 106, 12, 1, 100, 11, 106, 1, 111, 11,
106, 1, 103, 11, 106, 1, 115, 11, 106, 12, 1, 111, 11, 106, 1,
117, 11, 106, 1, 116, 11, 106, 12, 1, 32, 11, 106, 12, 8, 5, 12
```