Subroutines and loops in assembly

Revision: instructions so far:

opcode	mnemonic
1	Load X
2	Store X
3	Add X
4	Sust X
5	Input
6	Output
7	Halt
8	Skipcond → 000 - skip if AC<0; 400 - skip if AC=0; 800 - skip if AC>0
9	Jump X

Revision: if-else statements:

e.g. "if X>2 then Y=3 else Y=4"

becomes

```
        load X
        subt two
        skipcond 800    /skip if X>2
        jump else
if,     load three
        jump endif
else,   load four
endif,  store Y
```

Today's topics: (1) while loops
(2) some new instructions
(3) subroutines

(1) While loops (or equivalently, for loops)

These are easy using skipcond.

Example:

```
for counter = 1 to 5
    Y = Y + X
end
```

See loopDemo.mas for the assembly version

② New instructions

fill in this column interactively

[see table 4.6]

opcode	mnemonic	pseudocode
0	JnS X	M[X] = PC PC = X+1
A	Clear	AC = 0
B	AddI X	AC = AC + M[M[X]]
C	JumpI X	PC = M[X]
D	LoadI X	AC = M[M[X]]
E	StoreI X	M[M[X]] = AC

used for subroutines - see later in this lecture

Show demos of JnS, AddI, JumpI, Clear

See example 4.1 for a more practical use of AddI

③ Subroutines

subroutine (assembly) ≡ function (C, C++) ≡ method (Java)

Basic idea: - jump somewhere else, do some useful work, then return to where you were before

- the <u>return address</u> is the location to return to after the subroutine has finished its work.

- in MARIE, its best to store the return address at the start of the subroutine. (Use JnS for that)

- after doing the required job, use JumpI to return to the return address

example: see subroutineDemo.mas

Minilab:

(1) Step through subroutineDemo.mas, make sure you understand it.

(2) Alter subroutineDemo.mas to achieve the result

$$X = (4 * X - 1) * 4 * 4$$

(3) Write a subroutine that multiplies two positive integers X and Y, storing the result in Z.

(4) Use your answer to write an assembly program that computes the product of a list of five numbers

(5) Can one subroutine call another subroutine? Give examples or counterexamples.

(6) Can a subroutine call itself?