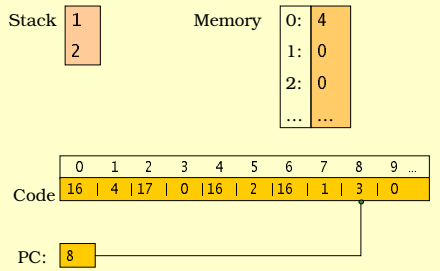
EVM

The Expressive Virtual Machine

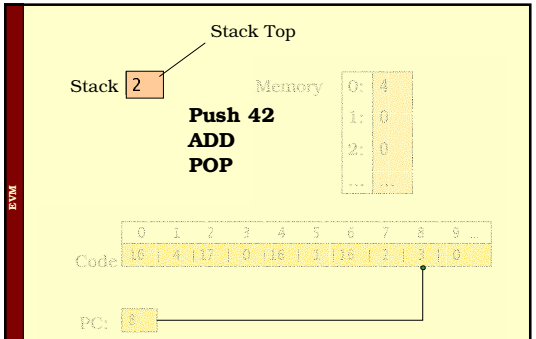
The EVM

- The EVM is a simple stack-machine.
- It consists of four components:
  1. A *Stack*.
  2. A *Memory* area.
  3. A *Code* area. (With a *program counter*, PC)
  4. A String pool.
- The elements stored in each position on the stack, in the memory and in the code are 32-bit integers (called *words*).
- The string pool is an array of strings.



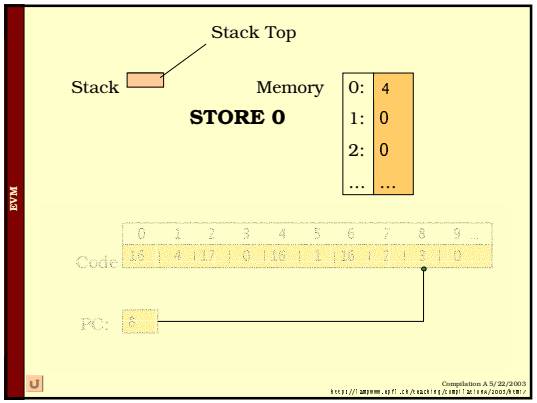
The Stack

- The stack is an *infinite* memory area on which words can be pushed to and popped from.
- All machine operations are performed on the words on the top of the stack and the results of each operations ends up on the stack.



The Memory

- The memory can theoretically contain 4,294,967,296 words.
- Each memory position can be referred to by its address, a number from 0 to 4,294,967,295 (i.e., a word).



The Code

- The code area is just another memory area with the same properties as the memory.
- There are no operations for writing to the code area.
- The program counter (PC) points to the next instruction to execute.
- Each instruction is coded with one or two words.

The Instructions

- The EVM has the following instructions:
  - Stack Operators:
    - PUSH X, POP, DUP
  - Arithmetic Operators:
    - ADD, SUB, MUL, DIV, REM, POW
  - Compare operators:
    - CMPL, CMPLE, CMPE
  - IO instructions:
    - PRINTI, PRINTB, PRINTS X, READI, READB
  - Memory operators:
    - STORE X, LOAD
  - Control flow instructions:
    - GOTO, JZ, JNZ, BREAK

Stack Operators

PUSH X
Stack: | S... | → | X | S... |
Memory: Unchanged
Program Counter: PC = PC + 2

POP
Stack: | X | → | S... | S... |
Memory: Unchanged
Program Counter: PC = PC + 1

DUP
Stack: | X | → | X | S... | X | S... |
Memory: Unchanged
Program Counter: PC = PC + 1

Arithmetic Operators

ADD
Stack: | y | x | S... | → | x+y | S... |
Memory: Unchanged
Program Counter: PC = PC + 1

SUB
Stack: | y | x | S... | → | x-y | S... |
Memory: Unchanged
Program Counter: PC = PC + 1

REM
Stack: | y | x | S... | → | x%y | S... |
Memory: Unchanged
Program Counter: PC = PC + 1

POW
Stack: | y | x | S... | → | x^y | S... |
Memory: Unchanged
Program Counter: PC = PC + 1

MUL
Stack: | y | x | S... | → | x*y | S... |
Memory: Unchanged
Program Counter: PC = PC + 1

DIV
Stack: | y | x | S... | → | x/y | S... |
Memory: Unchanged
Program Counter: PC = PC + 1

Compare operators

CMPL
Stack: | y | x | S... | → | 1 | S... | if x<y
| y | x | S... | → | 0 | S... | if x>=y
Memory: Unchanged
Program Counter: PC = PC + 1

CMPLE
Stack: | y | x | S... | → | 1 | S... | if x<=y
| y | x | S... | → | 0 | S... | if x>y
Memory: Unchanged
Program Counter: PC = PC + 1

CMPE
Stack: | y | x | S... | → | 1 | S... | if x==y
| y | x | S... | → | 0 | S... | if x<>y
Memory: Unchanged
Program Counter: PC = PC + 1

I/O Instructions

PRINTI
Stack:
| x | → | s… |
| s… |
Memory: Unchanged
Program Counter: PC = PC + 1
Side effect: prints x.

PRINTB
Stack:
| x | → | s… |
| s… |
Memory: Unchanged
Program Counter: PC = PC + 1
Side effect: prints *false* if x==0
otherwise prints *true*.

PRINTS x
Stack: Unchanged
Memory: Unchanged
Program Counter: PC = PC + 2
Side effect: Prints string x from the string pool.

READI
Stack:
| s… | → | x |
| s… |
Memory: Unchanged
Program Counter: PC = PC + 1
Side effect: reads x from stdin

READB
Stack:
| s… | → | x |
| s… |
Memory: Unchanged
Program Counter: PC = PC + 1
Side effect: reads x from stdin

Memory operators

STORE M
Stack:
| x | → | s… |
| s… |
Memory:
…	→	…
M:w		M:x
…		…
Program Counter: PC = PC + 2

LOAD M
Stack:
| s… | → | x |
| s… |
Memory:
| … | Unchanged
|M:x |
| … |
Program Counter: PC = PC + 1

Control Flow Instructions

GOTO L
Stack: Unchanged
Memory: Unchanged
Program Counter: PC = L

JZ L
Stack:
| x | → | s… |
| s… |
Memory: Unchanged
Program Counter: if x == 0 then PC = L else PC = PC + 2

JNZ L
Stack:
| x | → | s… |
| s… |
Memory: Unchanged
Program Counter: if x <> 0 then PC = L else PC := PC + 2

BREAK
Halts program execution.

Opcodes

BREAK	0	PRINTI	12
POP	1	PRINTB	13
DUP	2	READI	14
ADD	3	READB	15
SUB	4	PUSH	16
MUL	5	STORE	17
DIV	6	LOAD	18
REM	7	GOTO	19
POW	8	JZ	20
CMPL	9	JNZ	21
CMPLE	10	PRINTS	22
CMPE	11		

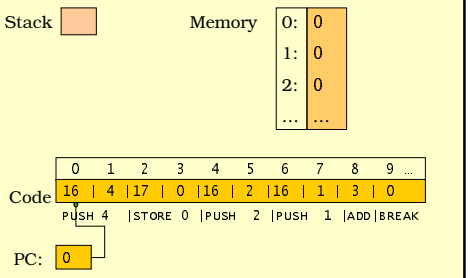
Example

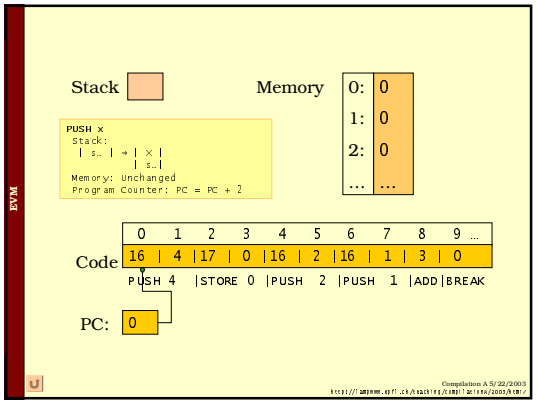
- Assembly Code:

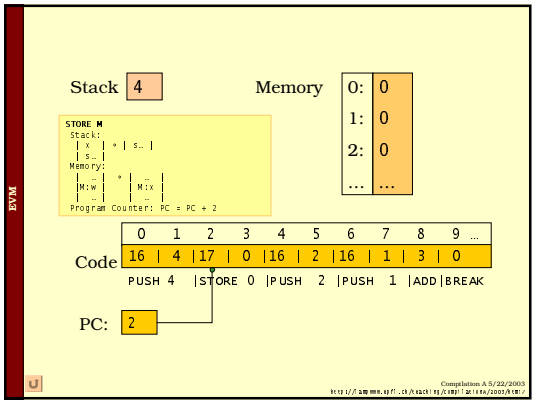
```
PUSH  4
STORE 0
PUSH  2
PUSH  1
ADD
BREAK
```

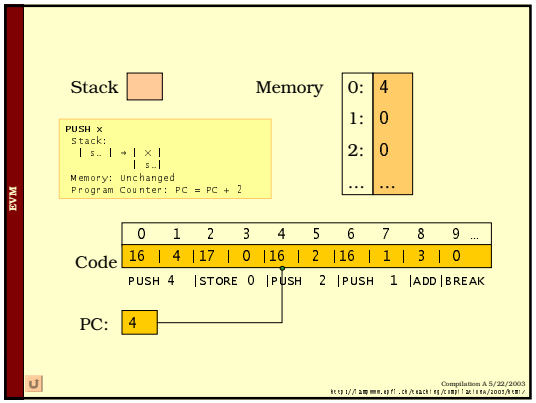
- Machine Code:

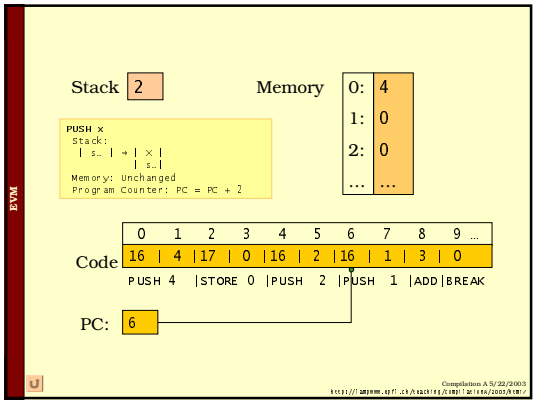
```
16    4
17    0
16    2
16    1
3
0
```

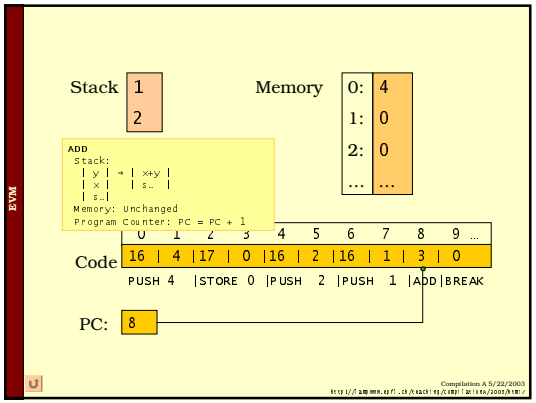


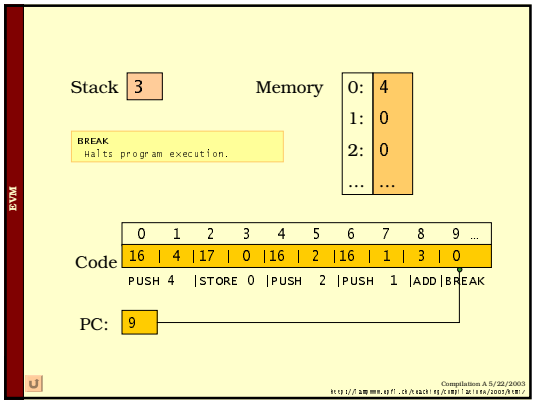

EVM
Stack
Memory
0: 0
1: 0
2: 0
PUSH x
Stack:
Memory: Unchanged
Program Counter: PC = PC + 2
0 1 2 3 4 5 6 7 8 9
Code
16 | 4 | 17 | 0 | 16 | 2 | 16 | 1 | 3 | 0
PUSH 4 | STORE 0 | PUSH 2 | PUSH 1 | ADD | BREAK
PC: 0
Compilation A 5/22/2003


EVM
Stack 4
Memory
0: 0
1: 0
2: 0
STORE M
Stack:
Memory:
Program Counter: PC = PC + 2
0 1 2 3 4 5 6 7 8 9
Code
16 | 4 | 17 | 0 | 16 | 2 | 16 | 1 | 3 | 0
PUSH 4 | STORE 0 | PUSH 2 | PUSH 1 | ADD | BREAK
PC: 2
Compilation A 5/22/2003


EVM
Stack
Memory
0: 4
1: 0
2: 0
PUSH x
Stack:
Memory: Unchanged
Program Counter: PC = PC + 2
0 1 2 3 4 5 6 7 8 9
Code
16 | 4 | 17 | 0 | 16 | 2 | 16 | 1 | 3 | 0
PUSH 4 | STORE 0 | PUSH 2 | PUSH 1 | ADD | BREAK
PC: 4
Compilation A 5/22/2003


EVM
Stack 2
Memory
0: 4
1: 0
2: 0
PUSH x
Stack:
Memory: Unchanged
Program Counter: PC = PC + 2
0 1 2 3 4 5 6 7 8 9
Code
16 | 4 | 17 | 0 | 16 | 2 | 16 | 1 | 3 | 0
PUSH 4 | STORE 0 | PUSH 2 | PUSH 1 | ADD | BREAK
PC: 6
Compilation A 5/22/2003


EVM
Stack 1
2
Memory
0: 4
1: 0
2: 0
ADD
Stack:
Memory: Unchanged
Program Counter: PC = PC + 1
0 1 2 3 4 5 6 7 8 9
Code
16 | 4 | 17 | 0 | 16 | 2 | 16 | 1 | 3 | 0
PUSH 4 | STORE 0 | PUSH 2 | PUSH 1 | ADD | BREAK
PC: 8
Compilation A 5/22/2003


EVM
Stack 3
Memory
0: 4
1: 0
2: 0
BREAK
Halts program execution.
0 1 2 3 4 5 6 7 8 9
Code
16 | 4 | 17 | 0 | 16 | 2 | 16 | 1 | 3 | 0
PUSH 4 | STORE 0 | PUSH 2 | PUSH 1 | ADD | BREAK
PC: 9
Compilation A 5/22/2003

EVM

EVM Assembler

- To make it easier to write programs for the EVM it has an assembler.
  - Positions in the code can be labeld (`LABEL n`).
  - Control flow instructions reffers to labels and not arbitrary positions in the code.
  - `PRINTS` can take a string argument.