

Stacks cont'd

Ata Kaban
University of Birmingham

Stacks for calculation

- Example:

$$(5 + 2) * \sqrt{x * x + y * y} + 8$$

What order are operations applied in?

$$(5 + 2) * \sqrt{x * x + y * y} + 8$$

1 6 5 2 4 3 7

$\sqrt{\quad}$ means
“square root of” (SQRT)
e.g.
SQRT($x * x + y * y$)

Reverse Polish notation

- Order of operations is as written
- No brackets needed
- Powerful use of stack (= operand stack) to store intermediate results

Number or variable: push on stack

Operation: apply to top elements on stack

E.g.

8	apply +	11
3		6
6		10
10		...
...		

$$(5+2)*\sqrt{x*x+y*y}+8$$

1 6 5 2 4 3 7

Reverse Polish: push operands, then operate.

We get:

$$5 \ 2 \ + \ x \ x \ * \ y \ y \ * \ + \ \sqrt{} \ * \ 8 \ +$$

1 2 3 4 5 6 7

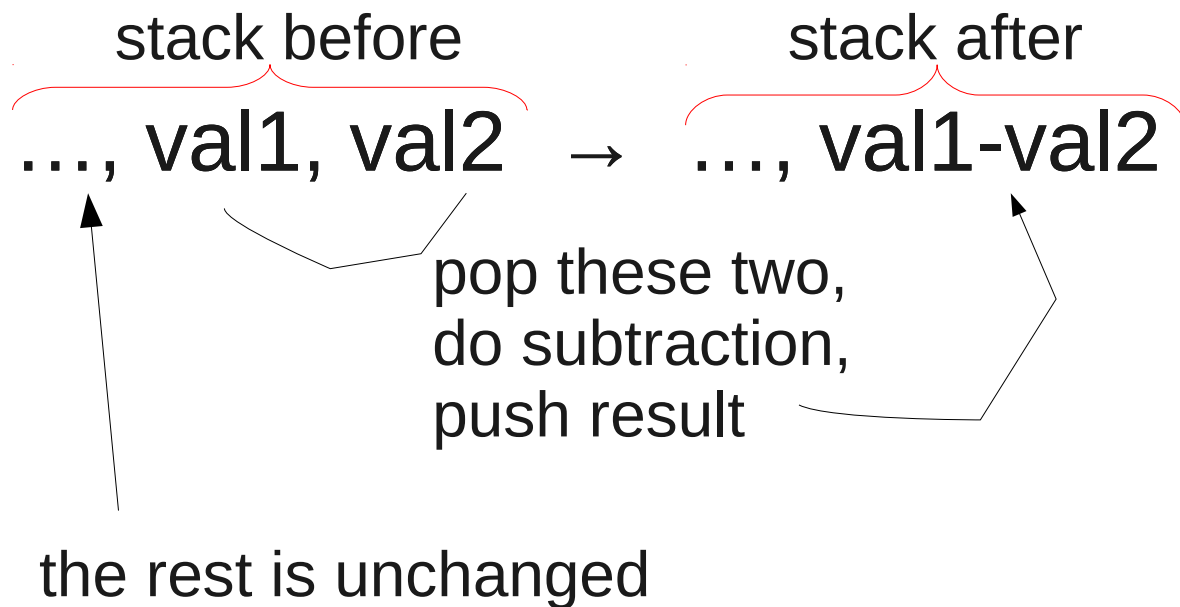
- Suppose that x has value 3, and y has value 4.
- Now, evaluate the expression. (Top of stack is on right.)

<u>Operation</u>	<u>Stack</u>
	empty
5	5
2	5,2
+	7
x	7,3
x	7,3,3
*	7,9
y	7,9,4
y	7,9,4,4
*	7,9,16
+	7,25
SQRT	7,5
*	35
8	35,8
+	43

Notation for operand stack

To show what an operation does to stack:

E.g. subtraction



Any expression can be converted to reverse Polish

- Then easy to execute with a stack
- Applications
 1. Humans use reverse Polish directly
 - E.g. some pocket calculator – HP in 1970s, 80s
 - Forth programming language has two stacks:
 - Operand stacks for calculations
 - Return stack for module calls

Applications of Reverse Polish

2. Compile to a reverse Polish form that is then executed.

- e.g. Postscript format, for printable files
 - executed by printers
- e.g. Java byte code
 - uses operand stacks for calculations
- In Java, each method call has its own operand stack.

Stack instead of registers

- Use 2 stacks
 - return stack for subroutine return
 - operand stack for reverse Polish calculations
- Don't need a,b,c registers
- Advantages:
 - More space for calculations
 - Opcodes don't need to specify registers
- Disadvantages:
 - Harder to know where things are on stack

What is an operand?

- Underlying meaning:
Whatever an operator operates on.
- Two meanings here (don't confuse them):
 - 1) Extra bytes after the instruction opcode in memory, e.g.

ld a 42

- 2) Entries in the operand stack.

Machine instructions as stack operators

[Forget the Toy CPU – remember these mnemonics (JVM)]

- Arithmetic:

e.g. **add** - adds top 2 stack entries

..., val1, val2 → ..., val1+val2

e.g. **sub** - subtracts top 2 stack entries

..., val1, val2 → ..., val1-val2

e.g. **neg** - negates top stack entry

..., val → ..., -val

- Similarity: **mul, div, rem**



remainder

Bitwise boolean operations

- Boolean operations on one bit
0=false, 1=true

OR	0	1
0	0	1
1	1	1

AND	0	1
0	0	0
1	0	1

XOR	0	1
0	0	1
1	1	0

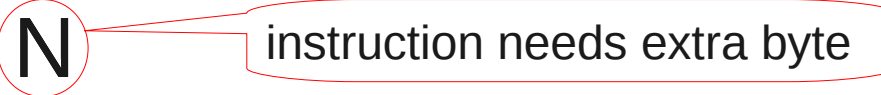
“eXclusive OR”

- Can do these bit-wise on binary values. Example for XOR:

```
0 0 1 1 1 0 1 1 1 1 ..... 0 1 1 0 0 1 0 0 0
0 0 0 1 0 0 1 0 0 1 ..... 1 1 1 1 0 1 1 0 0
-----
0 0 1 0 1 0 0 1 1 0 ..... 1 0 0 1 0 0 1 0 0
```

XOR: done on top 2 stack entries (similar to: or, and):
..., val1, val2 → ..., val1 XOR val2

Pushing constants on the stack

push N 
... → ..., N

-
- Pretend we also can push values of variables
[more on that next time]

load x
... → ..., value of x

Example: $5 \ 2 \ + \ x \ x \ * \ y \ y \ * \ + \ \sqrt{} \ * \ 8 \ +$

push 5
push 2
add
load x
load x
mul
load y
load y
mul
add
call $\sqrt{}$
mul
push 8
add

more
next time

<u>Operation</u>	<u>Stack</u>
	empty
5	5
2	5,2
+	7
x	7,3
x	7,3,3
*	7,9
y	7,9,4
y	7,9,4,4
*	7,9,16
+	7,25
SQRT	7,5
*	35
8	35,8
+	43

From slide 5

Jumps

- Unconditional jumps
 - Operand stack not used
- Conditional jumps

ifeq N // jumps to N if val=0

..., val → ...

if_cmpeq N // jumps to N if val1=val2

..., val1, val2 → ...



compare

Conditional jumps with other comparisons

jump if	val	$\left\{ \begin{array}{c} = \\ < \\ \leq \\ \neq \\ > \\ \geq \end{array} \right\}$	0	
				eq
				lt
				le
				ne
				gt
				ge

6 operators: ifeq, iflt, ifle, etc.
also: if_cmpeq, if_cmplt, etc.

Summary

You have now seen:

- Registers for calculating
- Operand stacks for calculating
- Return stacks for saving return addresses and registers

Next:

JVM puts them together: Bytecode

On to Exercise 3 and Exercise 4 on the 2-page Notes on Stacks (handed out yesterday).

- You will need to use an algorithm to convert math expressions from the usual “infix” notation to reverse-Polish. This algo is called Dijkstra's Shunting-Yard Algorithm.
 - The attachment explains how it works
 - You can also read the Wikipedia page on this algo
https://en.wikipedia.org/wiki/Shunting-yard_algorithm