

Consider

$$y = \frac{(x-1)(x-7)}{(x-11)}$$

With  $x = 10$  we may proceed as

- $(10-1) = 9$
- $(10-7) = 3$
- $(9*3) = 27$
- $(10-11) = -1$
- $27/(-1) = -27$

Writing  
intermediates  
on paper

## A Stack

- A Last In First Out (LIFO) data structure
- Consider a pile of paper we can
  - put a new piece of paper on the top
  - take the top piece of paper off
  - make the pile as high as we like (well almost!)
- Similarly for a data stack in the computer we can
  - *push* data onto the top of the stack
  - *pop* (take) data from the top of the stack
  - expand the stack to within the limits of the memory
- A convenient filling system for which we only need to remember the top position

## Arithmetic Logical Unit (ALU)

- Consider an ALU that given an operator *pops* the relevant number of data items from the stack and performs the operation placing the result back on the stack.
- *push 1, push 2, push 3, +, -*
- Write 1 on paper and put it on your desk
- Write 2 on paper and put it on top of the first paper
- Write 3 on paper and put on top of second paper
- Remove top two pieces of paper and add numbers (2+3), writing the result (5) on a new piece of paper that you put on the top of the paper pile
- Remove top two pieces of paper and subtract the numbers (1-5), write result (-4) on a new piece of paper that you put on your desk

## The HP Stack Calculator!

	HP Calculator	Stack (top .bottom)
	• 10 enter	• 10
	• 1	• 1, 10
	• -	• 9
Original	• 10 enter	• 10, 9
• (10-1) = 9	• 7	• 7, 10, 9
• (10-7) = 3	• -	• 3, 9
• (9*3) = 27	• *	• 27
• (10-11) = -1	• 10 enter	• 10, 27
• 27/(-1)=-27	• 11	• 11, 10, 27
	• -	• -1, 27
	• /	• -27

## Bit of a Pain!

- What if  $x=3.17284393124$ 
  - Do we really need to enter this number 3 times?
- NO, lets enter it once and store it in special memory (a “register”)

$$y = \frac{(x-1)(x-7)}{(x-11)}$$

- 3.17284393124 store 0
- 1 -
- Recall 0
- 7 -
- \*
- Recall 0
- 11 -
- /

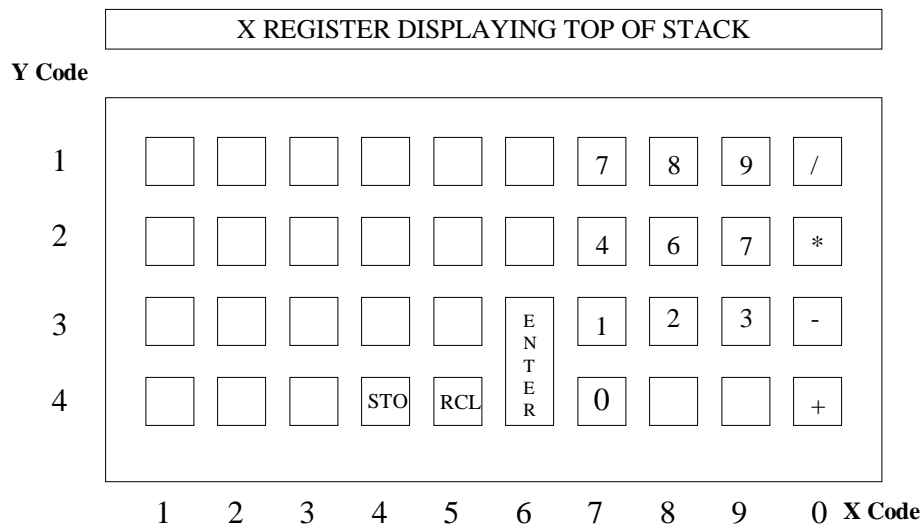
## Recap

- **Arithmetic Logical Unit:** performing arithmetic
- **Memory:** pieces of paper (maybe numbered) that we can write temporary results on and retrieve as required
- **Stack:** a convenient pile of paper for which memory addresses (numbers) are NOT needed
- **Registers:** useful places to store values that enter the computation in an unstructured way

## What If...

- We want to plot  $y$  for  $0 < x < 10$   
*Use a programmable calculator*
- We “program” the calculator
- Keystrokes are not executed, but a code representing each key is stored in memory together with any required data
  - Each memory location is given a number (or address) reflecting the order in which it was typed in
  - Like writing each of the keystrokes on a separate numbered piece of paper

## HP Machine Code (or an impressive PowerPoint picture)!



Address	Machine Code	Keystrokes	Comment
000	44 0	store 0	Store in register 0
002	1	1	Enter 1
003	30	-	Subtract
004	45 0	recall 0	Register 0 to stack
006	7	7	Enter 7
007	30	-	Subtract
008	20	*	Multiply
009	45 0	recall 0	Register 0 to stack
011	1	1	Enter 1
012	1	1	Make it 11
013	30	-	Subtract
014	10	/	Divide
015	43 32	g rtn	Return to calculator mode

## Machine Language Programming

- Now we can replace the function keypad with a simple numeric keyboard, with a computation defined as  
44 0 1 30 45 0 7 30 20 45 0 1 1 30 10 43 32
- Note
  - Instructions are 2 digits
  - Data items are single digits
- But remembering all the codes would be a major pain in the neck!

# An Easier Way

- Lets define a set of mnemonics

sto 44 0	rcl 45 0	div 10	mul 20
sub 30	add 40	ent 36	rtn 43 32

```

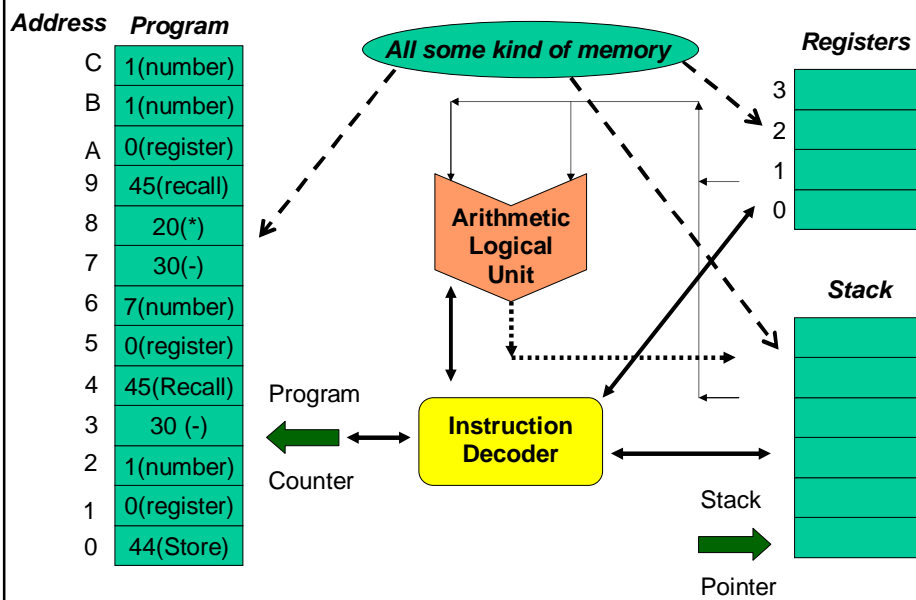
sto
1
sub
rcl
7
sub
mul
    
```

```

rcl
1
sub
div
rtn
    
```

This is ASSEMBLY language!

# Our Stack Calculator



## JVM – Java Virtual Machine

- Java bytecode is a stack machine code interpreter designed to execute Java bytecode
- Individual instructions are represented by numbers stored in a single byte

Mnemonic	Code	Args	Operation
iconst_0	3		push constant 0 onto stack
iconst_5	8		push constant 5 onto stack
iload_0	26		push local variable 0 onto stack
iload	21	n	push variable n onto the stack
istore_3	29		pop stack and store in local variable 3
pop	87		pop stack
iadd	96		pop elements, replace with sum
imul	104		pop elements, replace with product

## A Six Level Computer

(Tanenbaum Fig 1-2)

- Level 5: Problem solving language (eg C)
  - Compilation
- Level 4: Assembler
  - Translation
- Level 3: Operating system machine level
  - Partial interpretation
- Level 2: Instruction set architecture
  - Interpretation (microprogram) or direct execution
- Level 1: Microarchitecture level
  - Hardware
- Level 0: Digital logic level