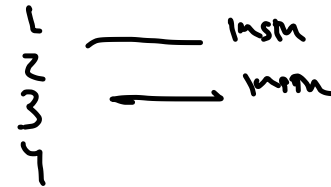


# RAM COMPUTER

- infinite # of registers  $\leftarrow$  <sup>memory</sup> holds arbitrary seq of symbols over some finite alph.
- instruction set

Initially all instructions for our program written in memory locations



## INSTRUCTION SET

<u>SUB</u>	ADD X, Y	ADD contents of reg X to contents of reg Y store result in reg X
	LOADC X, N	$x := N$ puts the const N in reg X
	COPY X, Y	gets value in reg Y, copies into reg X
	LOADI X, Y	$\cong$ to <u>COPY X contents(Y)</u>
	BRANCH X, Y	IF $\text{CONTENTS}(X) = 0$ THEN NEXT INSTR IS FOUND AT REG Y.
	HALT	
	plus whatever else you want.	

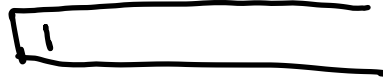
USE a 10-type TM

LOCATION TAPE



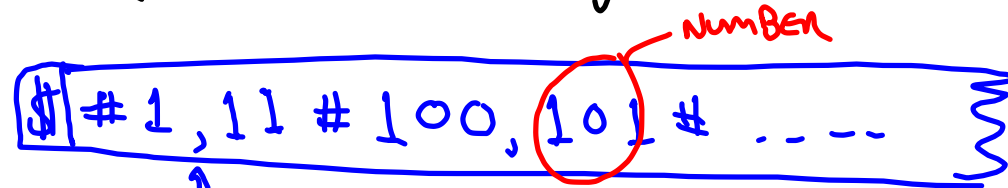
CURRENTLY EXECUTED RAM INSR  
IS IN REG 11

INITIALLY



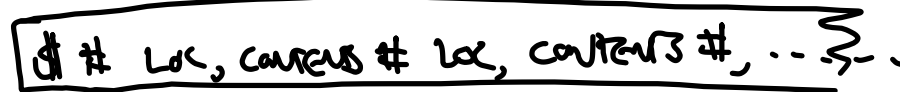
REGISTER TAPE

holds all register contents

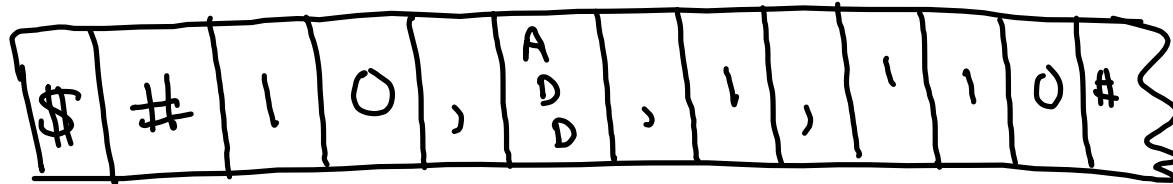


Reg 1 has "11"

REG 4 has "101"



INSTRUCTIONS ADD, LOADS, BRANCH, ...



# high-level view

- ① get next instruction ←
- ② execute it
- ③ update next-instruction-type

## SUBROUTINE ①

Resets head on Reg tape to left edge

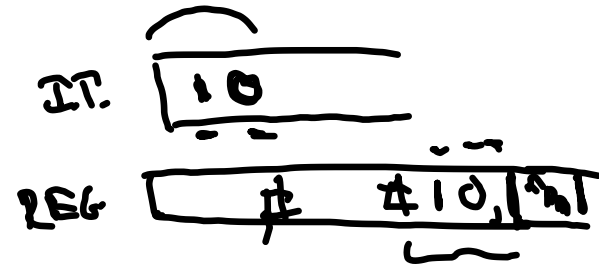
- DOES PATTERN MATCHING

FOR # P ,

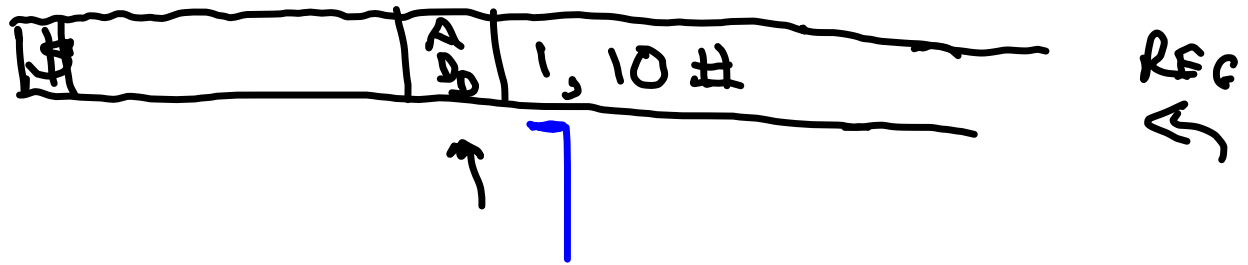
↑ what is on I.T.

- READS CHAR TELLING WHAT INSTR IS  
BRANCH TO APPROPRIATE TM CODE

INSTR TAP  
REG TAP



# EXAMPLE

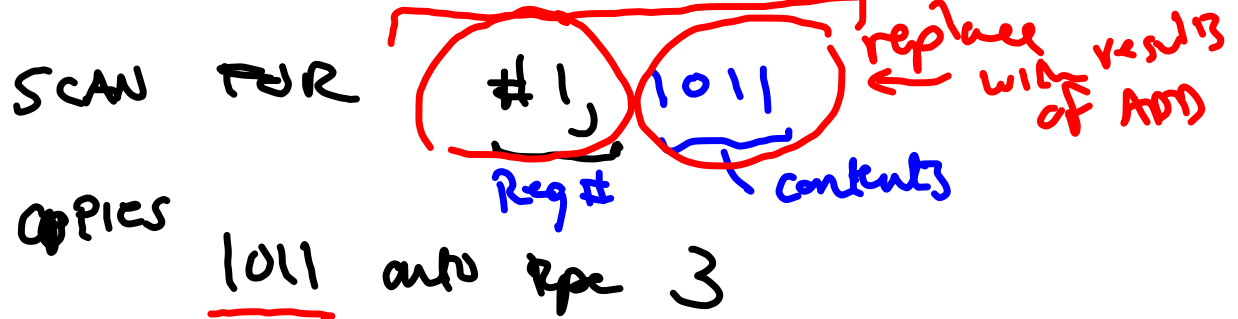


BRANCH TO  
M<sub>ADD</sub>

Copy next bits to two work tapes



- move REG TAPE HEAD TO LEFT TO #



- move Reg tape head to left to #



Thus, TMs can simulate RAMs

ONE TM " " ANY RAM

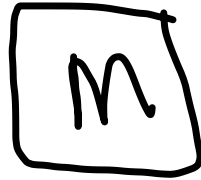
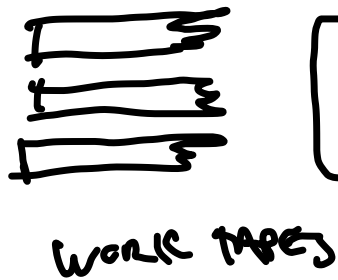
That TM can compute anything "algorithmic"

CHURCH - TURING THESIS

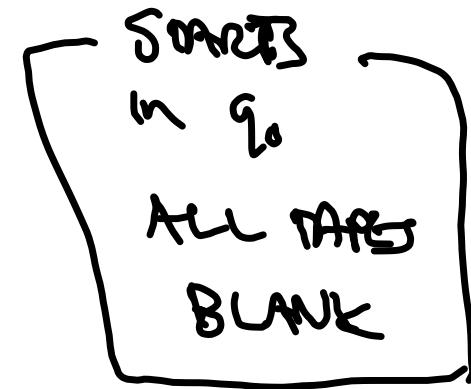
any  
mechanistic  
algorithmic  
computation

- ...
- CHURCH  $\lambda$ -calculus computable functions
  - RAM - machines
  - grammars / rewriting system / Markov ALGOS / semi-TM /  $\alpha \rightarrow \beta$
  - general recursive functions

# TMs as generators



WRITE-ONLY OUTPUT TAPE  
→ one way

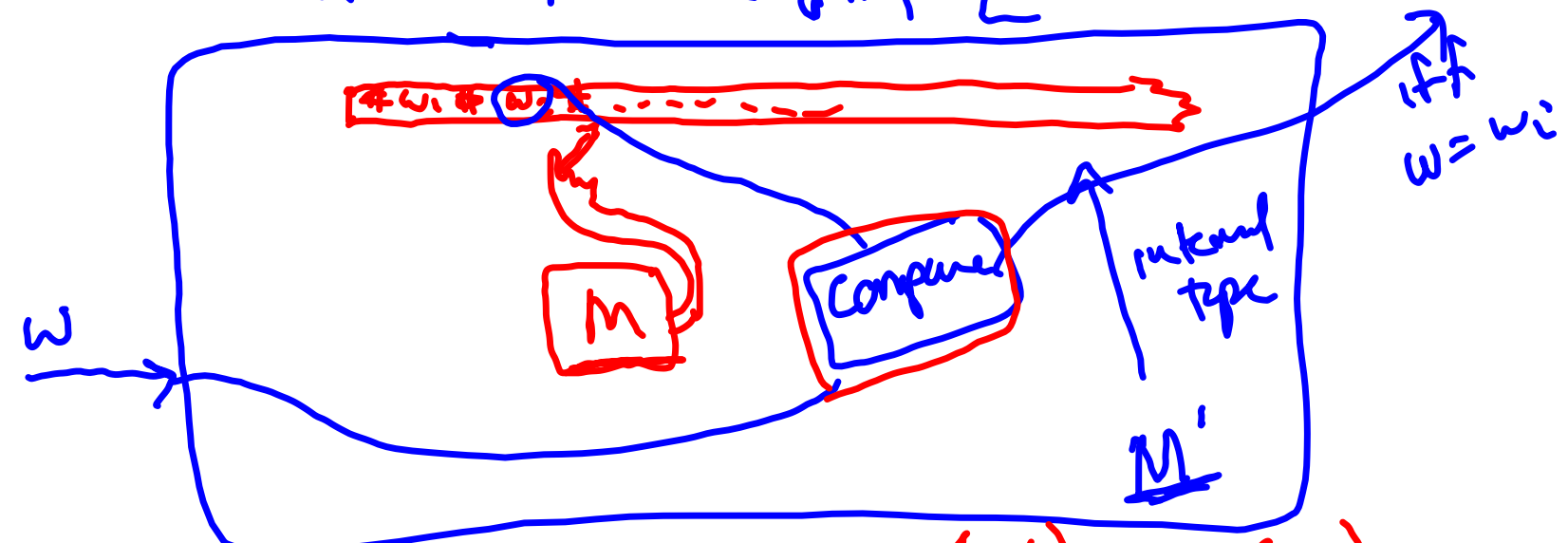


DEF  $G(M)$  "language generated by M"

$= \{ w : \#w\# \text{ appears on output tape} \}$

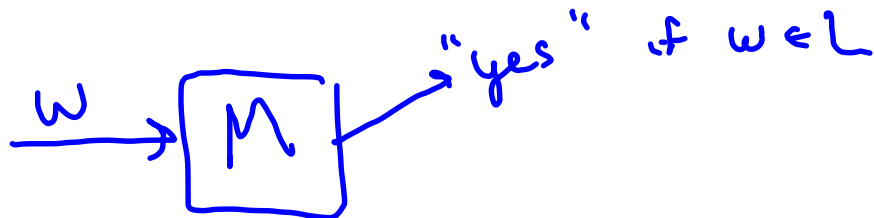
Theorem  $L$  is rec. enum. (i.e. accepted by some TM)  
IFF  $\exists M \quad L = \underline{G(M)}$

← Let  $M$  generate  $L$   
 construct  $M'$  accepting  $L$



generator  $\Rightarrow$  acceptor.  $L(M') = G(M)$

$\Rightarrow$  Suppose  $M$  accepts  $L$   
Construct  $M'$  generating  $L$



write #

For  $w = \epsilon$  to  $\dots \in \Sigma^*$

run  $M(w)$

if  $M(w)$  accepts

write  $w\#$  on tape.



DONE TAIL  
 $M$  on all  $w$   
simultaneously.