TURING MACHINE AS LANGUAGE ACCEPTORS

- A Turing machine halts when it no longer has available moves.
- If it halts in a final state, it accepts its input, otherwise it rejects its input.

For language accepted by M ,we define

L(M)={ $w \varepsilon \sum^+$: $q_0w \vdash x_1q_fx_2$ for some $q_f \varepsilon F$, x_1 ,$x_2 \varepsilon \Gamma^*$}

TURING MACHINE AS TRANSDUCERS

- To use a Turing machine as a transducer, treat the entire nonblank portion of the initial tape as input
- Treat the entire nonblank portion of the tape when the machine halts as output.

A Turing machine defines a function *y = f (x) for strings x, y* $\varepsilon \sum^*$ *if*

$q_0x \mid^* - q_f y$

- A function index is "Turing computable" if there exists a Turing machine that can perform the above task.

ID OF A TM

- Instantaneous Description or ID :
- $X_1 X_2 \ldots X_{i-1}$ **q** $X_i X_{i+1} \ldots X_n$ Means:

q is the current state

Tape head is pointing to Xi

X1X2…Xi-1XiXi+1… Xn are the current tape symbols

- δ (q , Xi) = (p ,Y , R) is same as:

$X_1 X_2 \ldots X_{i-1}$ **q** $X_i X_{i+1} \ldots X_n$ **|----** $X_1 X_2 \ldots X_{i-1}$ Y **p** $X_{i+1} \ldots X_n$

- δ (q Xi) = (p Y L) same as:

$X_1 X_2 \ldots X_{i-1}$ **q** $X_i X_{i+1} \ldots X_n$ **|----** $X_1 X_2 \ldots$ **p**X_{i-1}Y $X_{i+1} \ldots X_n$

TECHNIQUES FOR TM CONSTRUCTION

- Storage in the finite control
- Using multiple tracks
- Using Check off symbols
- Shifting over
- Implementing Subroutine

VARIATIONS OF TURING MACHINES

- Multitape Turing Machines
- Non deterministic Turing machines
- Multihead Turing Machines
- Off-line Turing machines
- Multidimensional Turing machines

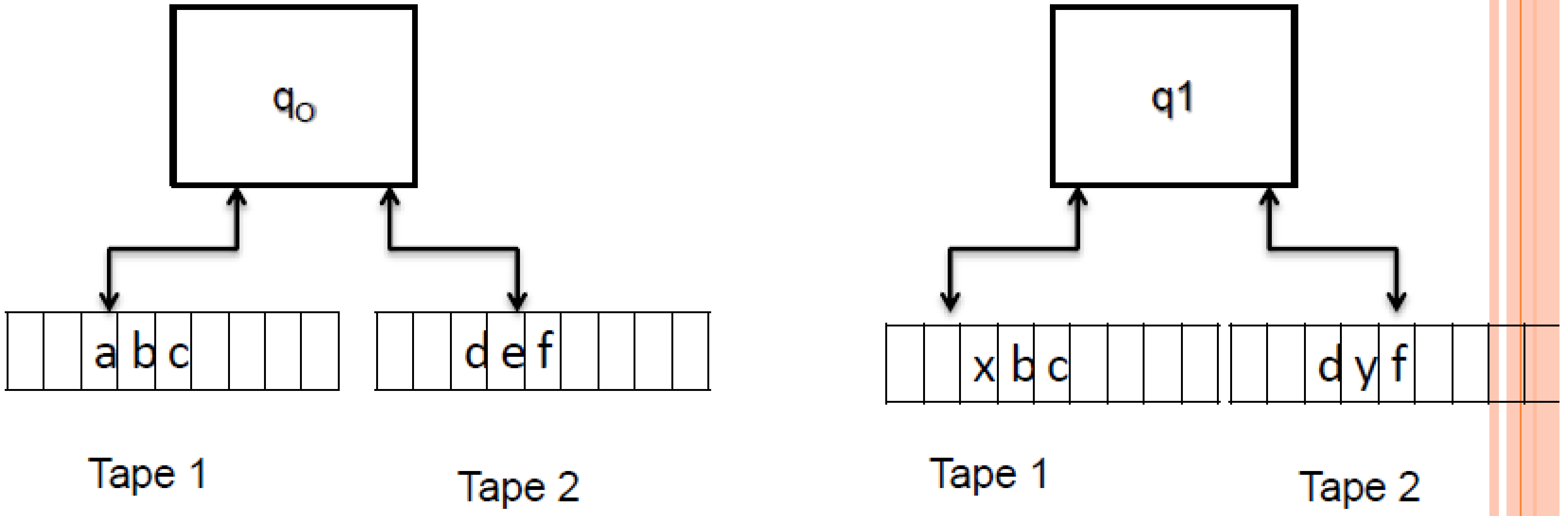
MULTITAPE TURING MACHINES

- A Turing Machine with several tapes
- Every Tape's have their Controlled own R/W Head
- For N- tape TM M=(Q,Σ, Γ,δ,q0,B,F)

we define $\delta : Q \times \Gamma^N \longrightarrow Q \times \Gamma^N \times \{ L , R\}^N$

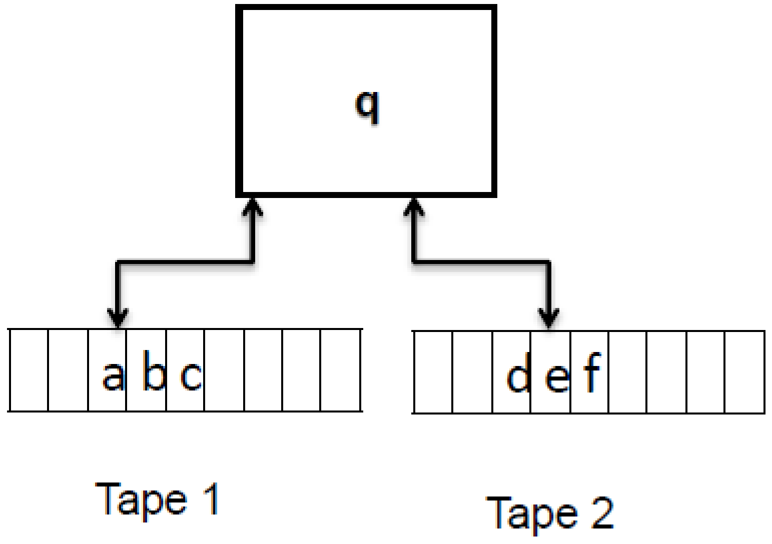
For e.g., if n=2 , with the current configuration

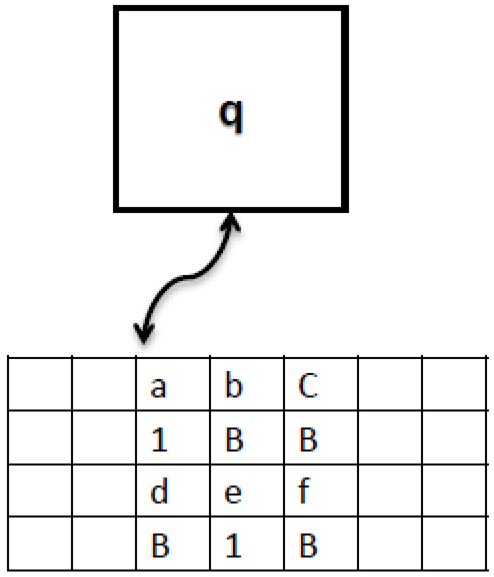
$\delta(q_O, a, e) = (q_1, x, y, L, R)$



SIMULATION

- Standard TM simulated by Multitape TM.
- Multitape TM simulated by Standard TM





NON DETERMINISTIC TURING MACHINES

- It is similar to DTM except that for any input symbol and current state it has a number of choices

- A string is accepted by a NDTM if there is a sequence of moves that leads to a final state

- The transaction function $\delta : Q \times \Gamma \longrightarrow 2^{Q \times \Gamma \times \{L, R\}}$

Simulation:

- A DTM simulated by NDTM

In straight forward way .

- A NDTM simulated by DTM

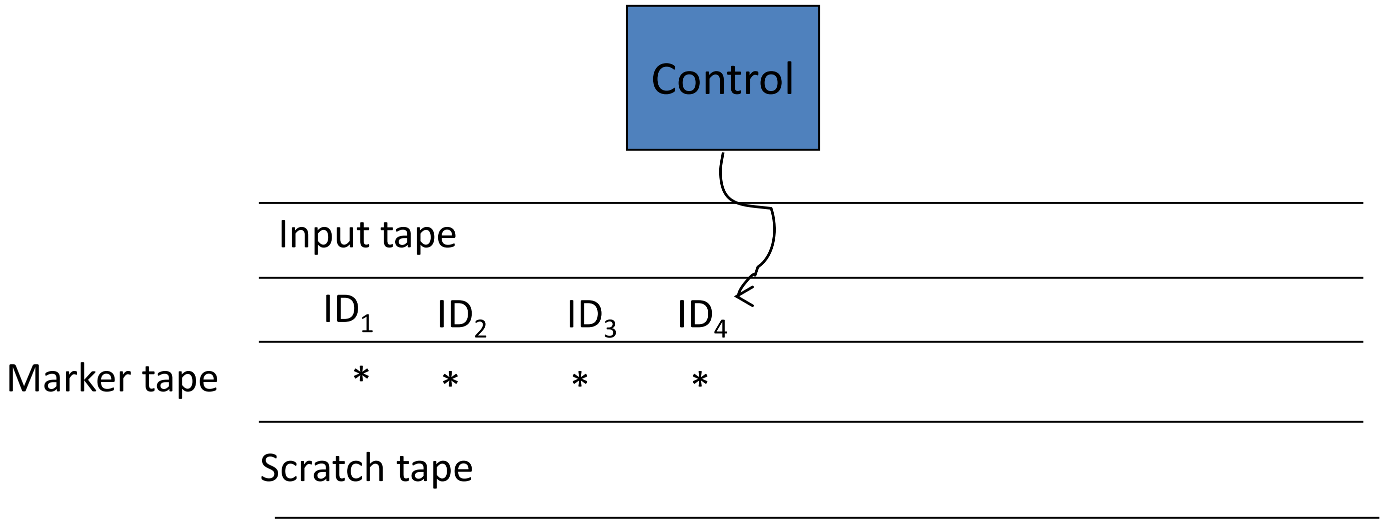
A NDTM can be seen as one that has the ability to replicate whenever is necessary

Non-deterministic TMs

A TM can have non-deterministic moves:

$\delta(q,X) = \{ (q_1,Y_1,D_1), (q_2,Y_2,D_2), \dots \}$

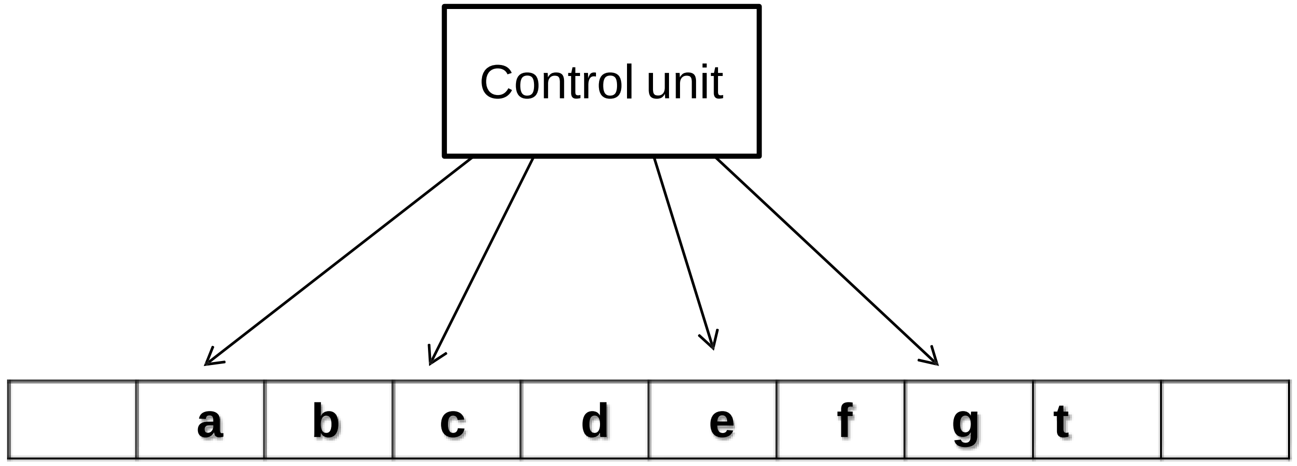
Simulation using a multitape deterministic TM:



MULTIHEAD TURING MACHINE

- Multihead TM has a number of heads instead of one.

- Each head indepently read/ write symbols and move left / right or keep stationery.



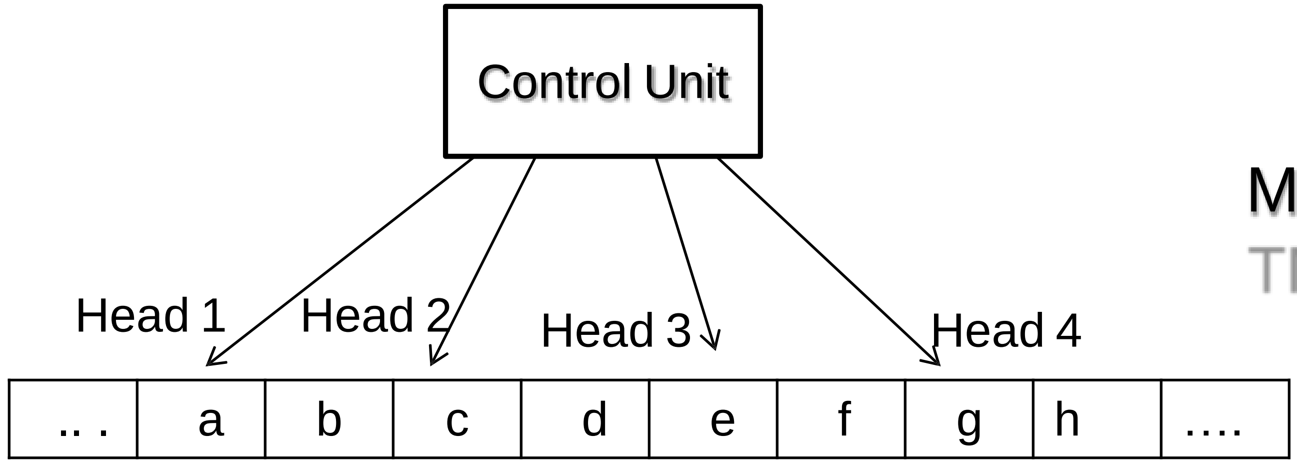
SIMULATION

- Standard TM simulated by Multihead TM.

  - Making on head active and ignore remaining head

- Multihead TM simulated by standard TM.

  - For k heads Using (k+1) tracks if there is..



Multihead TMTM

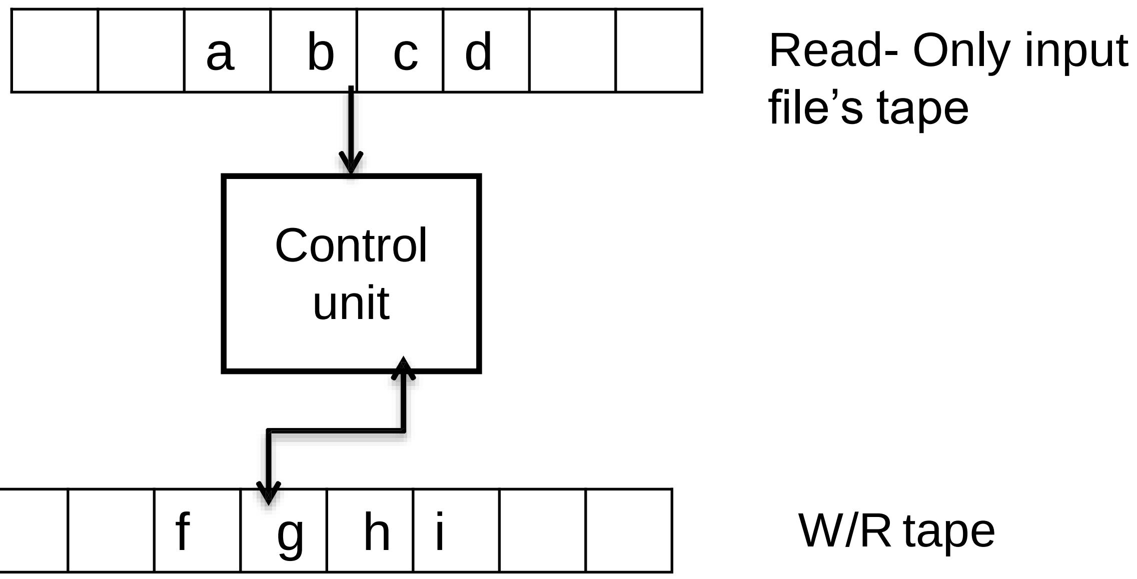
1st track	….	1	B	B	B	B	B	B	B	..
2nd track	….	B	B	1	B	B	B	B	B	..
3rd track	..	B	B	B	B	1	B	B	B	..
4th track	..	B	B	B	B	B	B	1	B	.
5th track	..	a	b	c	d	e	f	g	h	.

Multi track TM

OFF- LINE TURING MACHINE

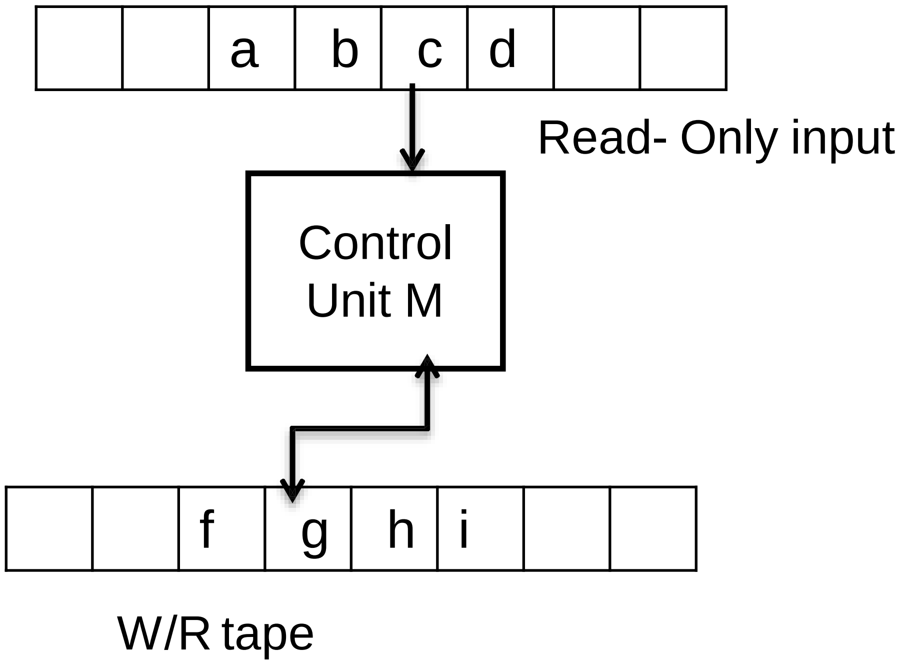
- An *Offline Turing Machine has two tapes*

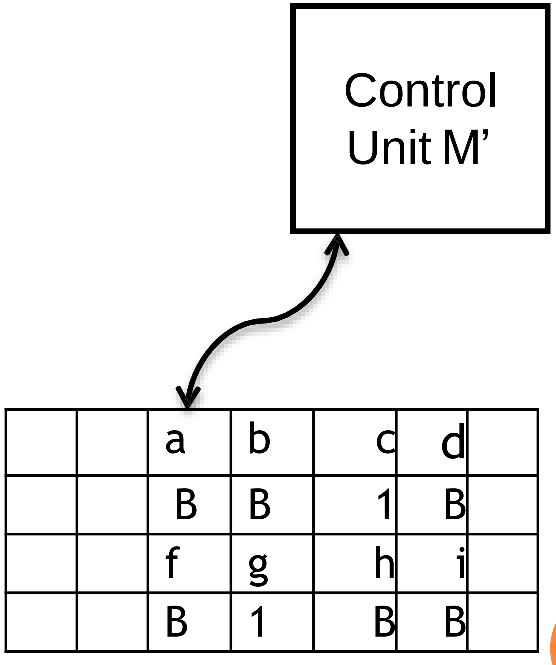
1. *One tape is read-only* and contains the input
2. The other is read-write and is initially blank.



SIMULATION

- A Standard TM simulated by Off-line TM
- An Off- line TM simulated by Standard TM





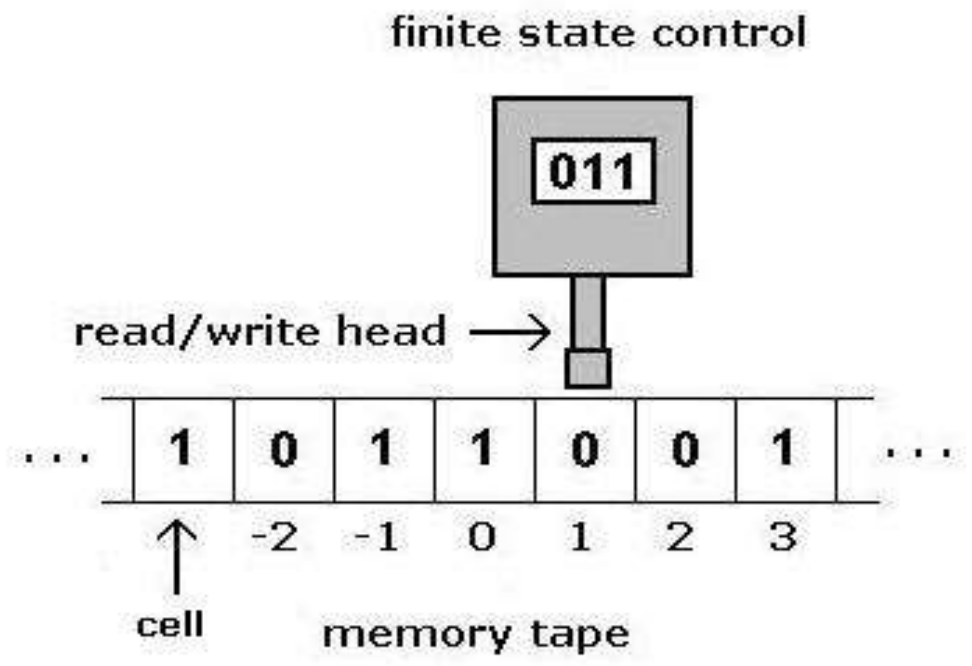
Restricted Turing Machines

- TM with semi infinite tape
- Multi stack TM
- Counter Machines

The Turing Machine

- **Turing machines** are extremely basic abstract symbol-manipulating devices which, despite their simplicity, can be adapted to simulate the logic of any computer that could possibly be constructed.
- They were described in 1936 by Alan Turing.
- Though they were intended to be technically feasible, Turing machines were not meant to be a practical computing technology, but a thought experiment about the limits of mechanical computation; thus they were not actually constructed

The Turing Machine



The Turing Machine

- Each move by a *Turing Machine* results in
- Change of state;
- Writing a tape symbol in the cell just scanned; and
- Moving the tape head left or right.

The Turing Machine

- Hopcroft and Ullman define a one-tape Turing machine formula: M=(Q, Γ, b, ∑, δ, q_0, F) where:

- *Q* is a finite set of *states*
- Γ is a finite set of the *tape alphabet/symbols*
- b is the *blank symbol* (the only symbol allowed to occur on the tape infinitely often at any step during the computation) - Delimiter
- Σ, a subset of Γ not including b is the set of *input symbols*
- δ = Q x Γ → Q x Γ x {L,R} is a partial function called the *transition function*, where L is left shift, R is right shift.
- q_0 is the *initial state*
- F is the set of *final* or *accepting states*

Restricted Turing Machines

- **Semi-infinite Tapes' TM**

- A TM with a ***semi-infinite*** tape means that there are no cells to the left of the initial head position.

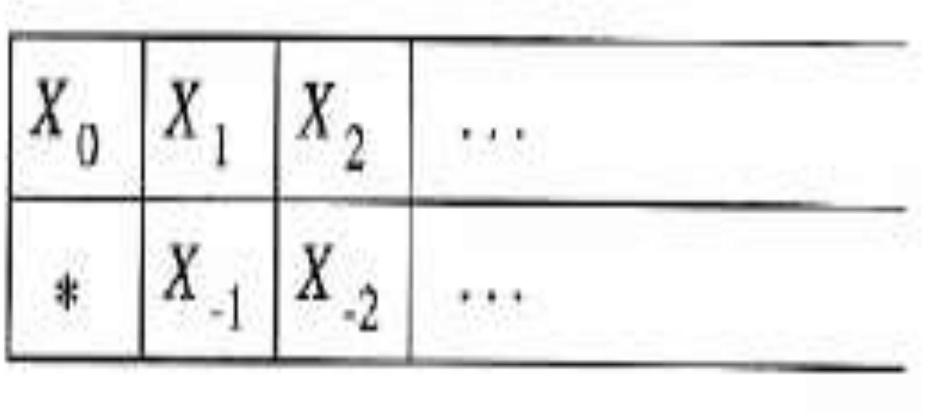
- A TM with a semi-infinite tape simulates a TM with an infinite tape by using a two-track tape.

Semi-infinite Tapes TM

- The use of two tracks is as follows:
- The upper track represents the cells of the original TM that are at the right of the initial head position.
- The lower track represents the cells at the left of the initial head position, but in reverse order.

Semi-infinite Tapes TM

A semi-infinite tape that apparently can simulate a two-way tape:



X_0	X_1	X_2	...
*	X_{-1}	X_{-2}	...

Semi-infinite Tapes TM

- **Theorem 8.12**

- Every language accepted by a TM *M*2 is also accepted by a TM *M*1 with the following restrictions:
  - *M*1's head never moves left of its initial position; and
  - *M*1 never writes a blank.

Multistack Machines

- Generalizations of the PDAs
- TMs can accept languages that are not accepted by any PDA with one stack.
- But PDA with two stacks can accept any language that a TM can accept.

Multistack Machines

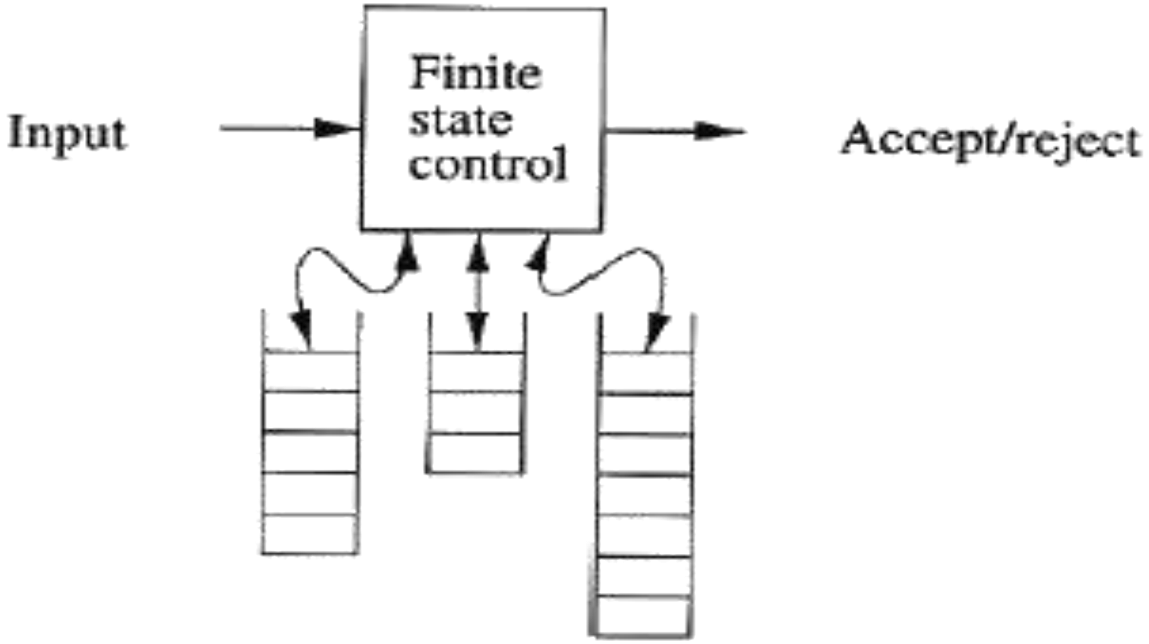


Figure 8.20: A machine with three stacks

Multistack Machines

- A k-stack machine is a deterministic PDA with k stacks.
- It obtains its input from an input source rather than having the input placed in a tape.
- It has a finite control, which is in one of a finite set of states.
- It has a finite stack alphabet, which it uses for all its stacks.

Multistack Machines

- A move of a multistack machine is based on:
  - The state of the finite control
  - The input symbol read, which is chosen from the finite input alphabet
  - The top stack symbol on each stack

Multistack Machines

- In one move:
  - a ***multistack machine*** can change to a new state $q \in Q$;
  - and replace the top symbol of each stack with a string of zero or more stack symbols $X \in \Gamma^*$.

Multistack Machines

- The typical transition function of k-stack machine looks similar to:

$$\delta(q, a, X_1, X_2, \ldots, X_k) = (p, \gamma_1, \gamma_2, \ldots, \gamma_k)$$

In state q, with X_i on top of ith stack,

for i= 1, 2, …, k, the machine may consume input a, go to state p, and replace X_i on top of the ith stack by string y_i, for i=1,2, …, k.

Multistack Machines

- To make it easier for a multistack machine to simulate a TM, we introduce a special symbol called the endmarker, represented by $.
- The role of the endmarker is To let us know when we have consumed all the available input.
- The endmarker appears only at the end of the input and is not part of it.

Multistack Machines

- Theorem 8.13

- If a language L is accepted by a turing machine, L is accepted by a two-stack machine

Multistack Machines

- Proof
- The idea is that one stack can hold what is to the left of the head, while the other holds what is to the right of the head, neglecting all the infinite blank symbols beyond the leftmost and rightmost of the head.

Multistack Machines

- Proof Cont’d
- Let’s have a two-stack machine S, simulating a one-tape TM M.
- S begins with a bottom-of-stack marker on each stack, considered to be as a start symbol for the stacks, and must not appear elsewhere on the stacks

Multistack Machines

- Proof Cont’d
- Suppose that w$ is on the input of S.
- S copies the input w onto its first stack, and ceases to copy when reading the endmarker on the input.
- S pops each symbol in turn from its first stack and pushes it onto its second stack.

Multistack Machines

- Proof Cont’d
- The first stack of S is empty.
- The second stack holds w, with the left end of w is at the top.
- S simulates the start state of M , “empty cells at the left of the head”, and S has a second stack holding w, representing the fact that w appears at and to the right of the M’s head.

Multistack Machines

- Proof Cont'd
- S simulates a move of M as follows:

1. S knows the state of M, say q, because S simulates the state of M in its own finite control.
2. S knows the symbol X scanned by the head of M; it's at the top of the second stack. If the second stack contains only the endmarker, this means that M's head has just scanned a blank.

Multistack Machines

- Proof Cont'd

3. So, S knows the next move of M.
4. The next state of M is recorded in a component of S's finite control, in place of the previous state.
5. If M replaces X by Y and moves right, then S pushes Y onto its first stack, representing that Y is now at the left of M's head.

Multistack Machines

- Proof Cont'd

6. If M replace X by Y and moves left, S pops the top of the first stack , say Z, then replaces X by ZY on the second stack. This represents what used to be one position left of the head is now at the head.

   If Z is the bottom-of-stack marker, then S must push BY onto the second stack and not pop the first stack.

Multistack Machines

- Proof Cont'd

7. S accepts if the new state of M is accepting.

   Otherwise, S simulates another move of M in the same manner.

Counter Machines

- Two equivalent ways to think of a ***counter machine***:

- A stack with a bottom marker, say Z_0, and one other symbol, say X, that can be placed on the stack. Thus, the stack always looks like:

  $XXX\ldots XZ_0$, or specifically, X^nZ_0.

- A device that holds a non-negative integer with operations *increment-by-1*, *decrement-by-1*, and *test-if-zero*.

Counter Machines

- **The Power of Counter Machines**

- ***Every language accepted by a counter machine is recursively enumerable.***
  - Counter machines are special cases of stack machines, which are special cases of multitape TMs, which accept only recursively enumerable languages.

- ***Every language accepted by a one-counter machine is a CFL.***
  - A one-counter machine is a special case of a onestack machine; *i.e.*, a PDA.

Counter Machines

- **Theorem 8.14**

- Every recursively enumerable language is accepted by a three-counter machine.

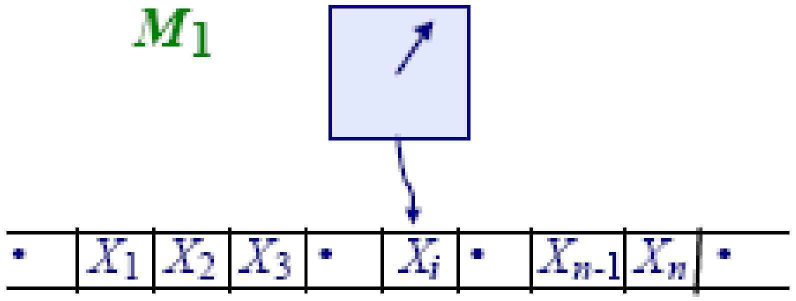
Counter Machines

- Proof

- Idea: Use Theorem 8.13 to derive a two-stack machine, then develop a constructive algorithm for a *2-stacks-to-3-counters* conversion.

Counter Machines

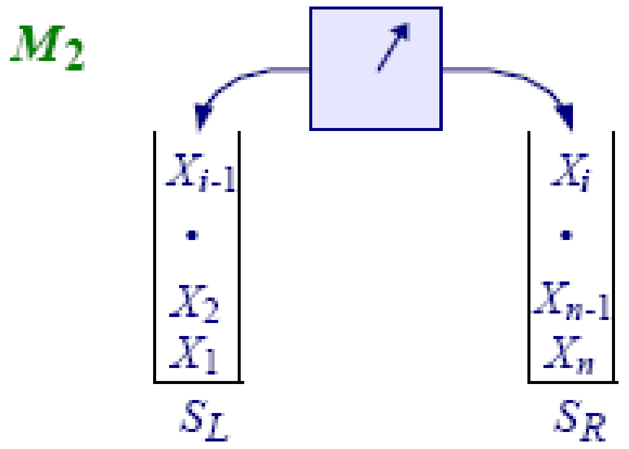
- Proof Cont'd
- ***2-stacks-to-3-counters* conversion:**
- Consider $L=\{a^n b^n c^n : n >= 0\}$, TM ***M*1** where $L($**M1**$)=L$, and w є $\{a,b,c\}^*$ where $|w|=n$.



- So, by Theorem 8.13 we can derive an equivalent 2-stack machine, ***M2*** ...

Counter Machines

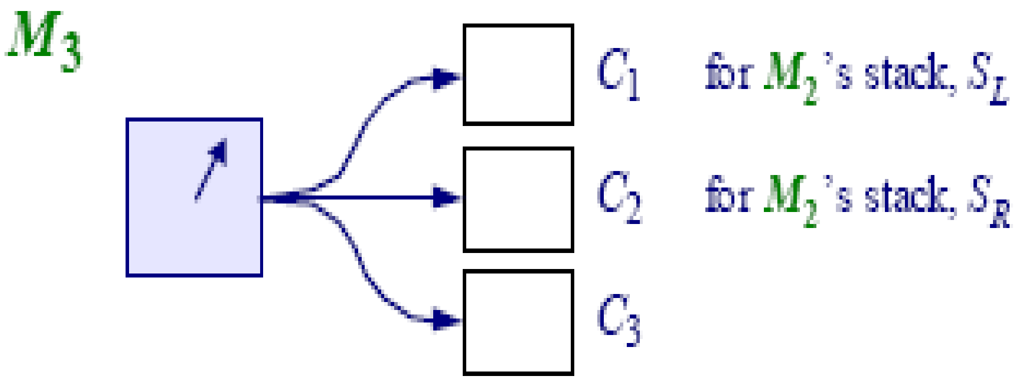
- Proof Cont'd



- If a stack has *r*-1 symbols, think of the stack contents as a base-*r* number with the symbols as the digits 1 through *r*-1.

Counter Machines

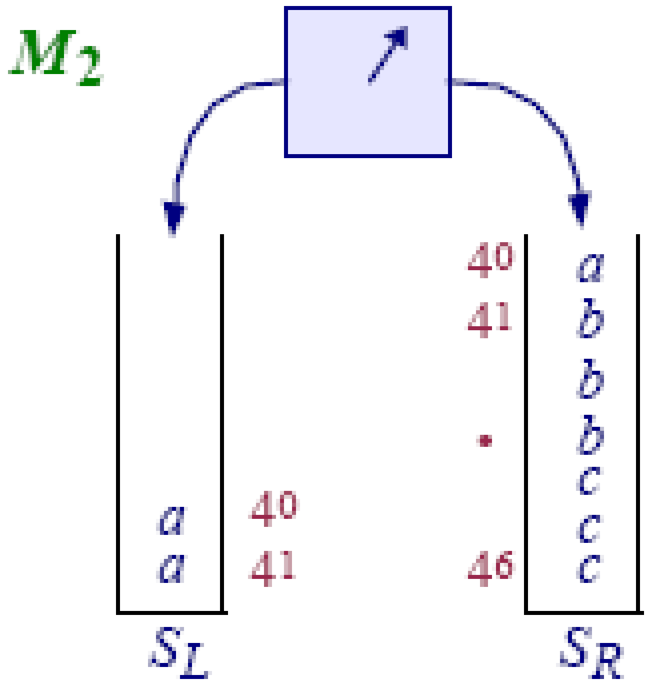
- Proof Cont'd
- So, for the 3-counter machine, ***M3***, use one counter for each stack plus one "scratch" counter:



M_3
C_1 for M_2's stack, S_L
C_2 for M_2's stack, S_R
C_3

Counter Machines

- Proof Cont'd

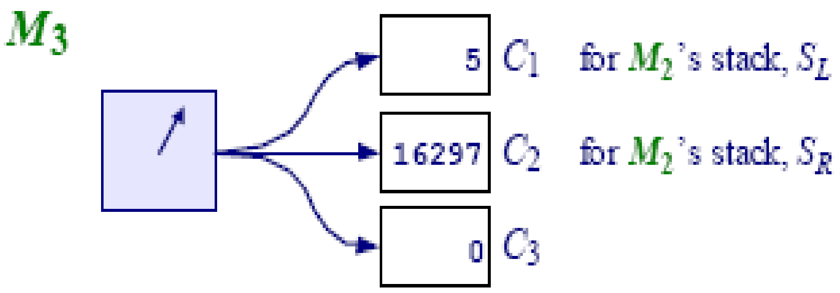


There are r-1=|{a,b,c}|=3 stack symbols. So, r=4.

Let a=1, b=2, c=3 represent the three (non-zero) base-4 quad units.

Counter Machines

- Proof Cont'd

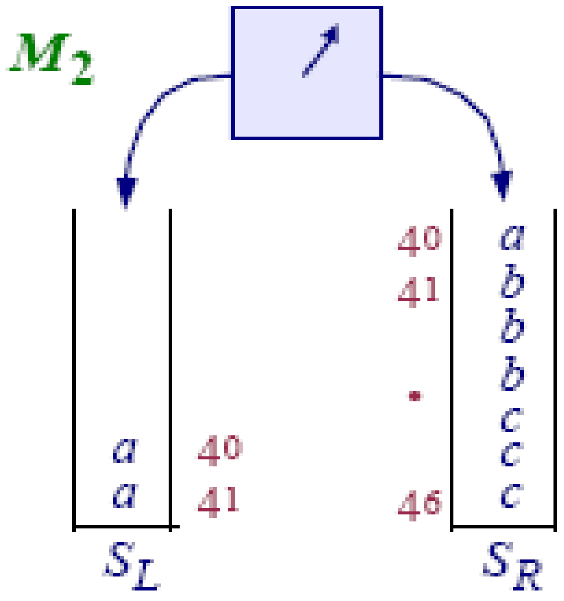


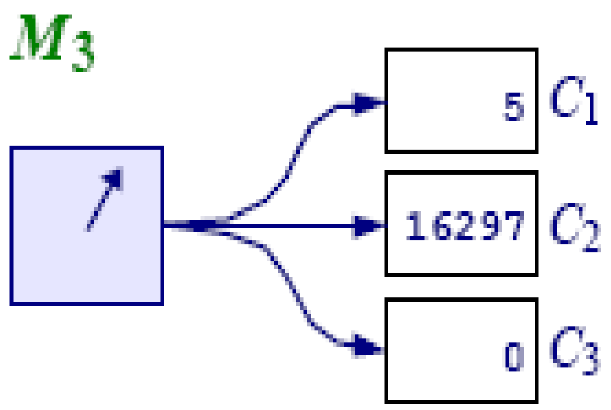
M_3
5
16297
0
C_1 for M_2's stack, S_L
C_2 for M_2's stack, S_R
C_3

- Note: To move ***M*1**'s read-write head one cell to the right requires popping X_i from S_R and pushing X_i into S_L for ***M*2**.

Counter Machines

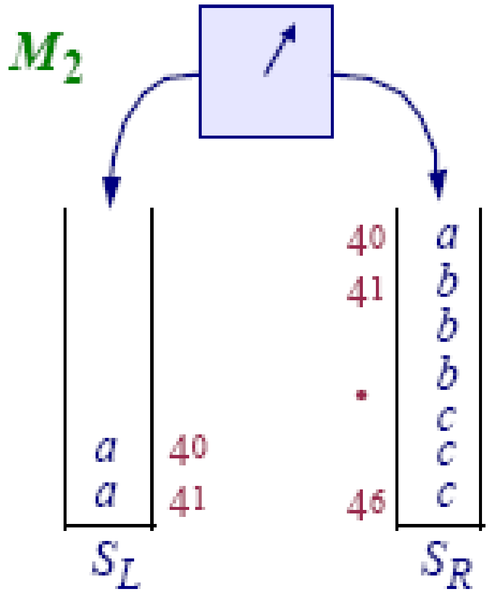
- Proof Cont'd
- Consider the move from *aaqabbbccc* to *aaarbbbccc* in ***M*1** where *q*,*r* ϵ *Q* ...

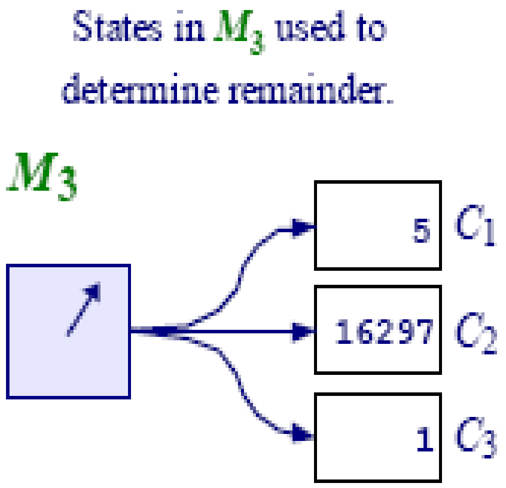




Counter Machines

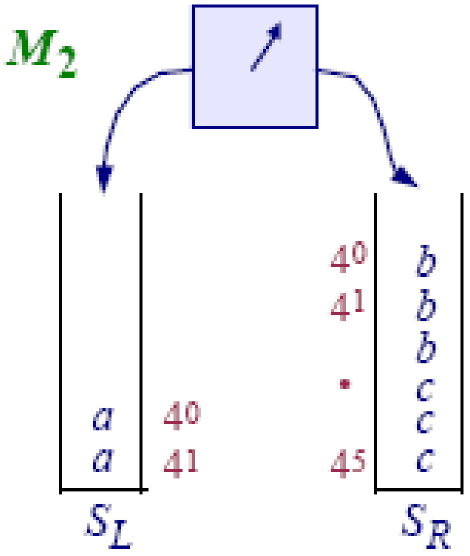
- Proof Cont'd
- Read top symbol of S_R = store the remainder, C_2 modulo r, into C_3.

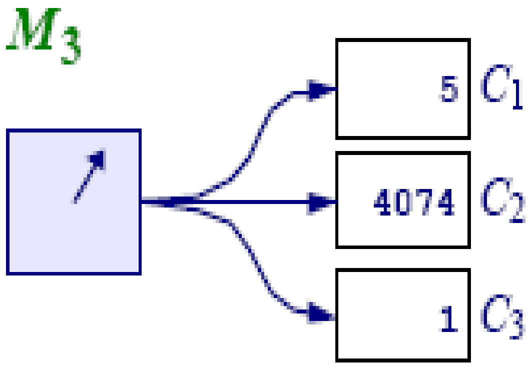




Counter Machines

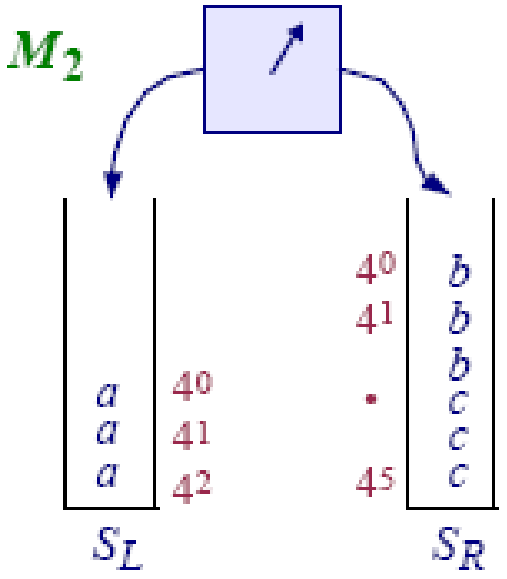
- Proof Cont'd
- Pop from S_R = divide C_2 by r, discarding the remainder.

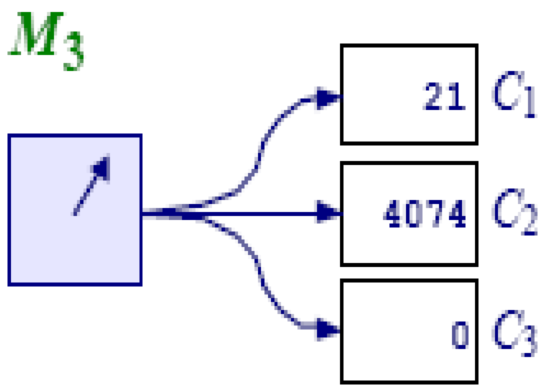




Counter Machines

- Proof Cont'd
- Push into S_L = multiply C_1 by r, then add the value stored in C_3.





Counter Machines

- Conclusion:

- Two counters are used to hold the integers that represent each of the two stacks.

- The third counter is used to adjust the other two counters. In particular, to perform the division or multiplication of a count by r.

Counter Machines

- **Theorem 8.15**

- Every recursively enumerable language is accepted by a two-counter machine.

Counter Machines

- Proof

- Idea: Develop a constructive algorithm for a *3-counters-to-2-counters* conversion. Do this by representing the 3 counters *i*, *j*, and *k*, by a single integer $m=2^i\,3^j\,5^k$ *counter, and a second counter that help multiplying or dividing m by one of the three primes 2, 3, and 5*.

Counter Machines

- Proof cont'd

- Store the number $m=2^i \, 3^j \, 5^k$ in one counter and use the other one as a "scratch" counter.

- Test if $i=0$ by moving count from one counter to the "scratch" counter, counting modulo 2 in the state.

- $i=0$ if and only if the number $m=2^i \, 3^j \, 5^k$ is not divisible by 2.

- tests for $j=0$ and $k=0$ analogous.

Counter Machines

- Proof cont'd

- Incrementing counters *i*, *j*, or *k* is equivalent to multiplications of the count $m=2^i 3^j 5^k$ by 2, 3, or 5; respectively.

- Decrementing counters *i*, *j*, or *k* is equivalent to divisions of the count $m=2^i 3^j 5^k$ by 2, 3, or 5; respectively.

TMs and Computers

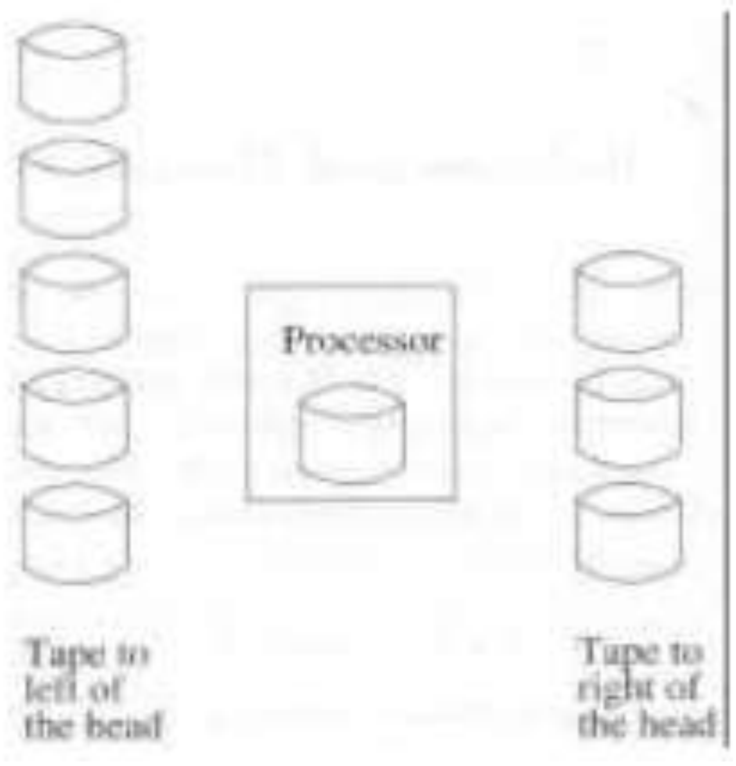
- In one sense, a (real) computer has a finite number of states, and is thus *weaker* than a TM .
- We have to postulate an infinite supply of tapes, disks, or some peripheral storage device to simulate an infinite tape TM.
- Assume a human operator can mount disks, keeping them stacked neatly on the sides of the computer.

TMs and Computers

- **Simulating a TM by a Computer**
- A computer can simulate finite control, and mount one disk that holds the region of the tape around the tape head.
- When the tape head moves off this region, the computer outputs the following request:
  - Move the current disk to the top of the left or right pile; and
  - Mount the disk at the top of the other pile.

TMs and Computers

- **Simulating a TM by a Computer**



TMs and Computers

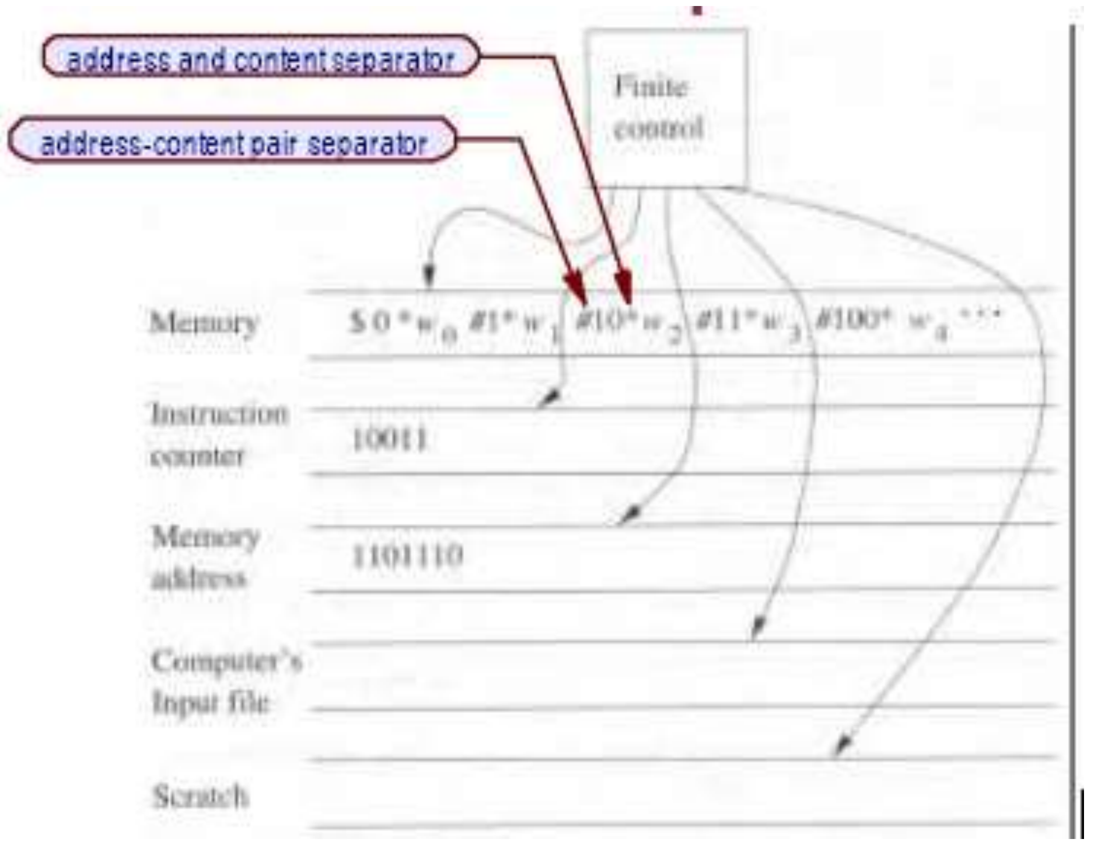
- **Simulating a Computer by a TM**
- Idea: Simulation is at the level of stored instructions and words in memory .
- TM has one tape that holds all the used memory locations and their contents.
- Other TM tapes hold the instruction counter, memory address, computer input file, and "scratch."

TMs and Computers

- **Simulating a Computer by a TM**
- Instruction cycle of computer simulated by:
  - Find the word indicated by the instruction counter on the memory tape.
  - Examine the instruction code (a finite set of options), and get the contents of any memory words mentioned in the instruction, using the “scratch” tape.
  - Perform the instruction, changing word values as needed, and adding new address,value pairs to the memory tape, if needed.

TMs and Computers

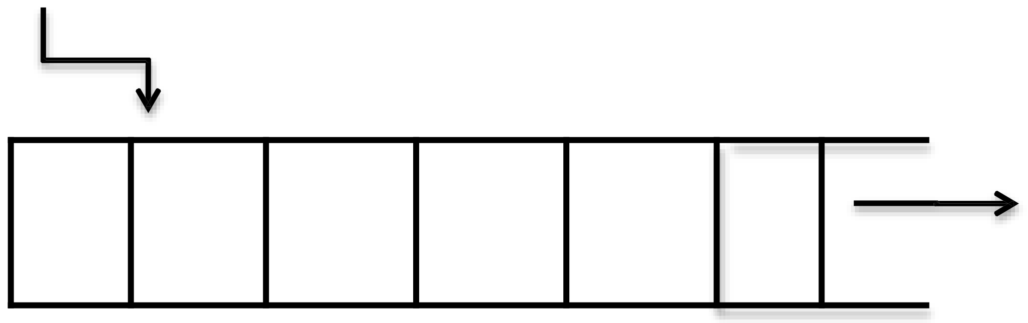
- **Simulating a Computer by a TM**


address and content separator
address-content pair separator
Finite control
Memory
$ 0 * w0 #1* w1 #10* w2 #11* w3 #100* w4 ...
Instruction counter
10011
Memory address
1101110
Computer's Input file
Scratch

TURING MACHINE WITH SEMI- INFINITE TAPE

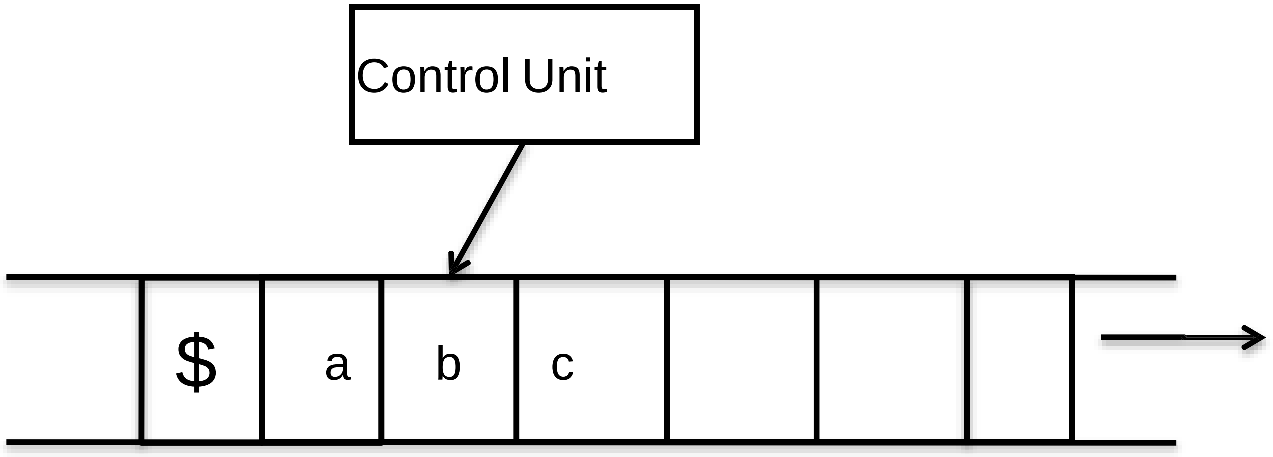
- A Turing machine may have a “*semi-infinite tape”, the nonblank* input is at the extreme left end of the tape.

- Turing machines with semi-infinite tape are equivalent to Standard Turing machines.

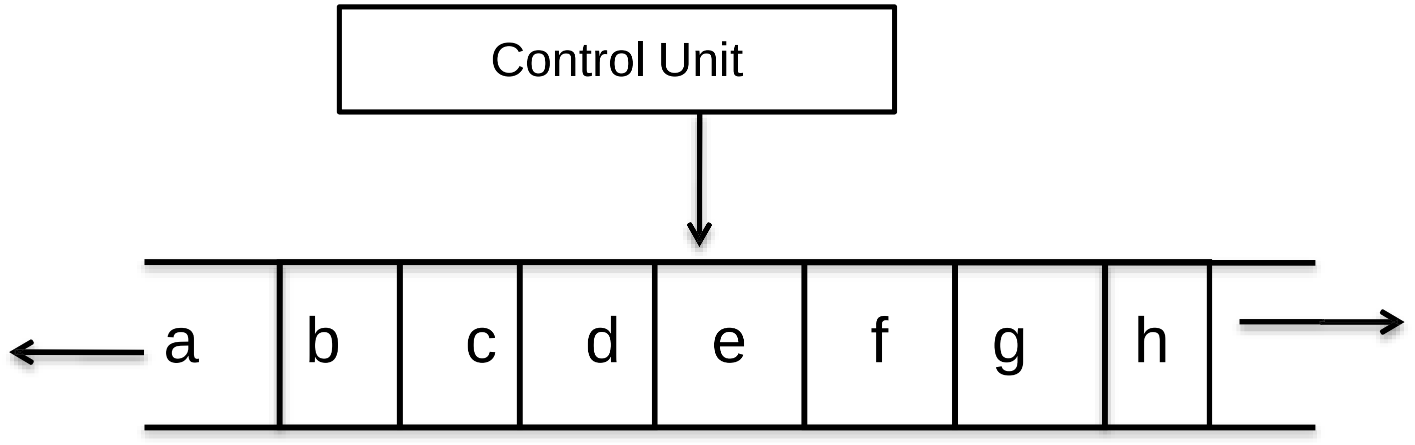


SIMULATION

- Semi – infinite tape simulated by two way infinite tape



- Two way infinite tape simulated by semi -infinite tape


Control Unit
a
b
c
d
e
f
g
h