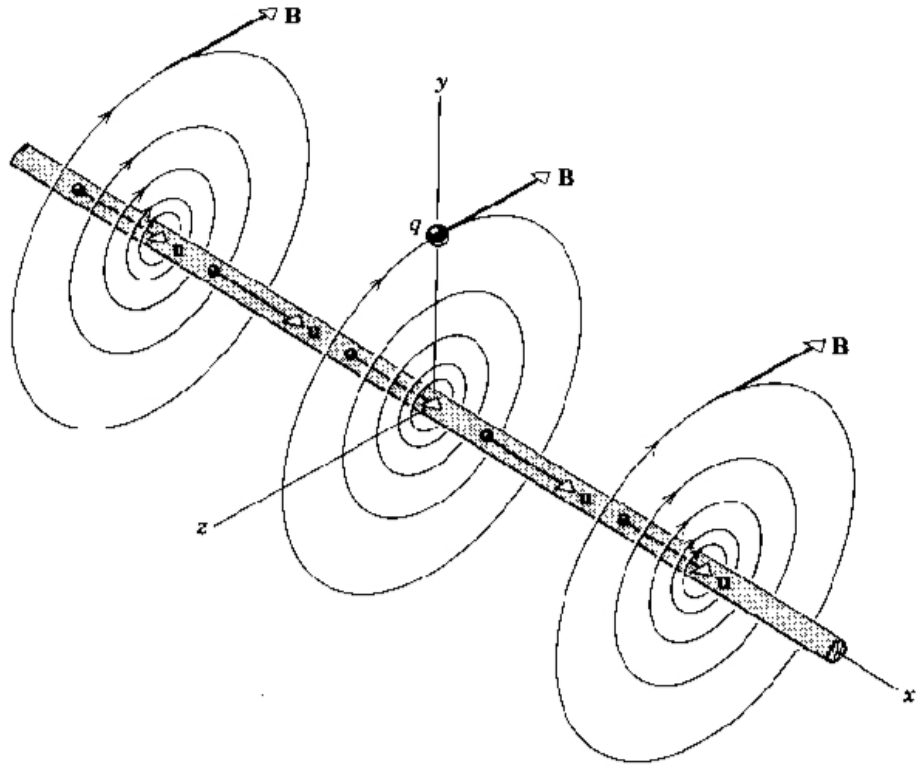
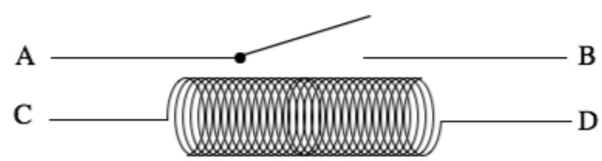
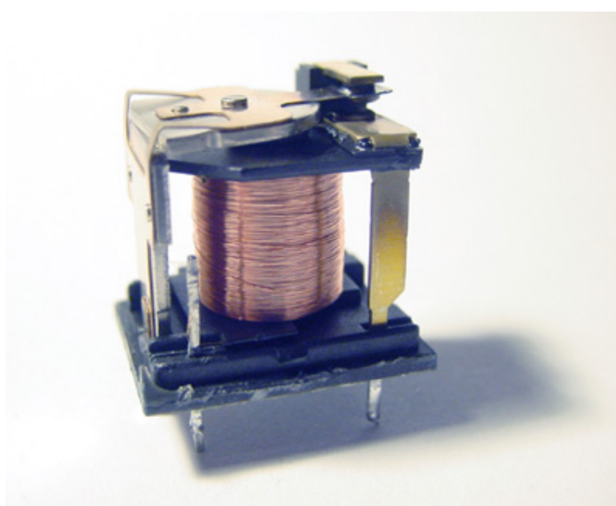
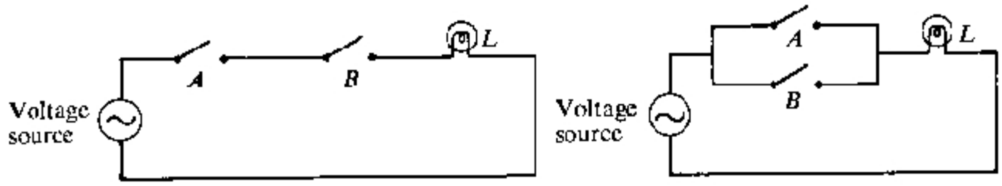

B
B
B
y
q
z
x

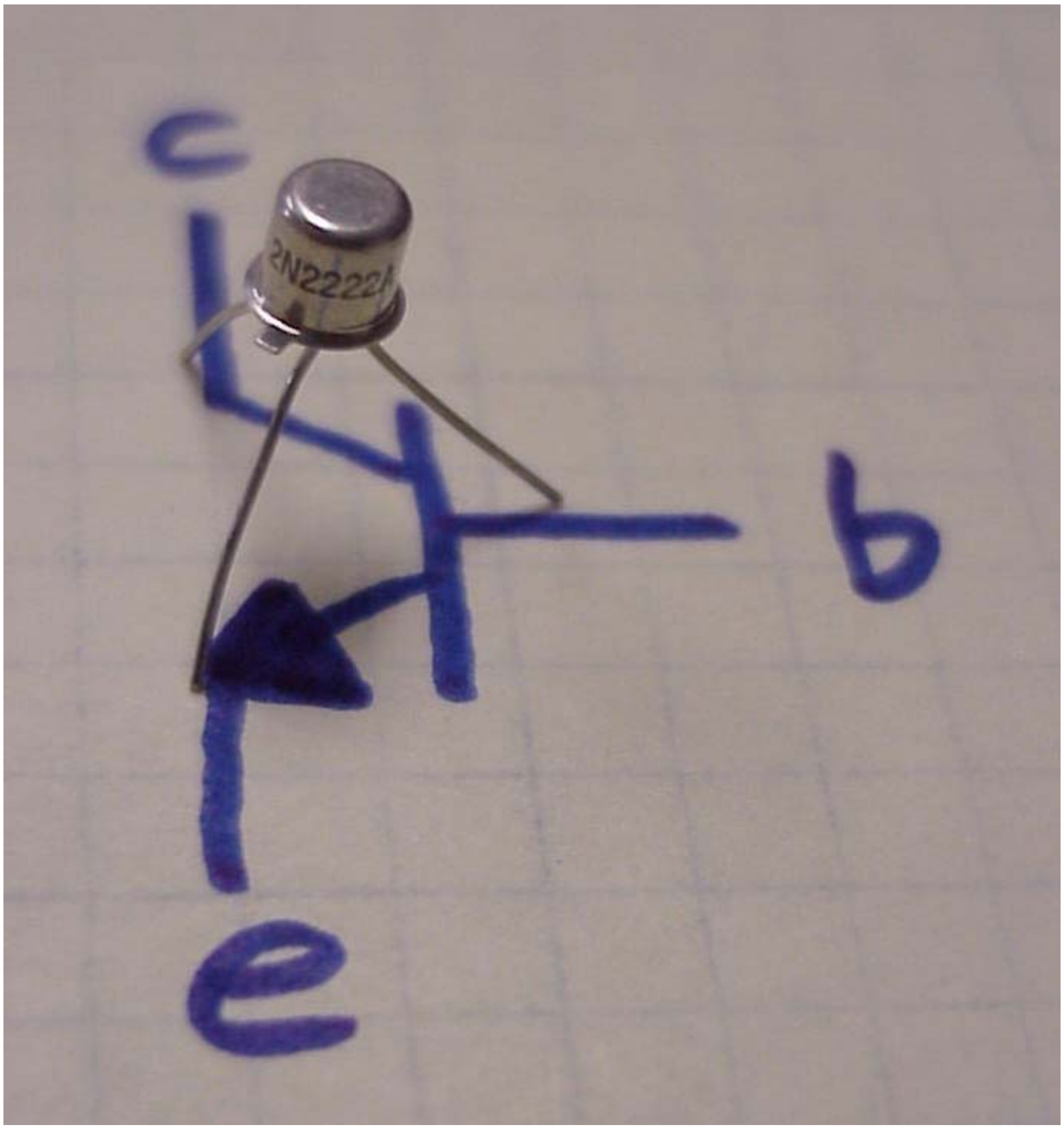

A
B
C
D

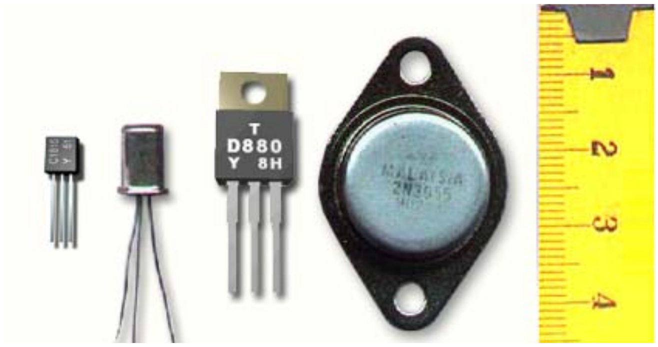


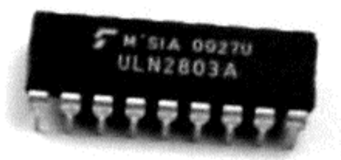


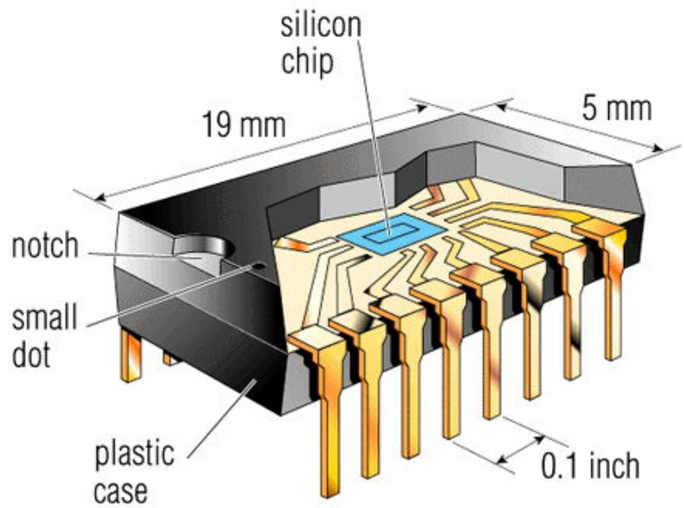
(a) Switches in series – logic AND (b) Switches in parallel – logic OR

Figure 1-4 Switching circuits that demonstrate binary logic

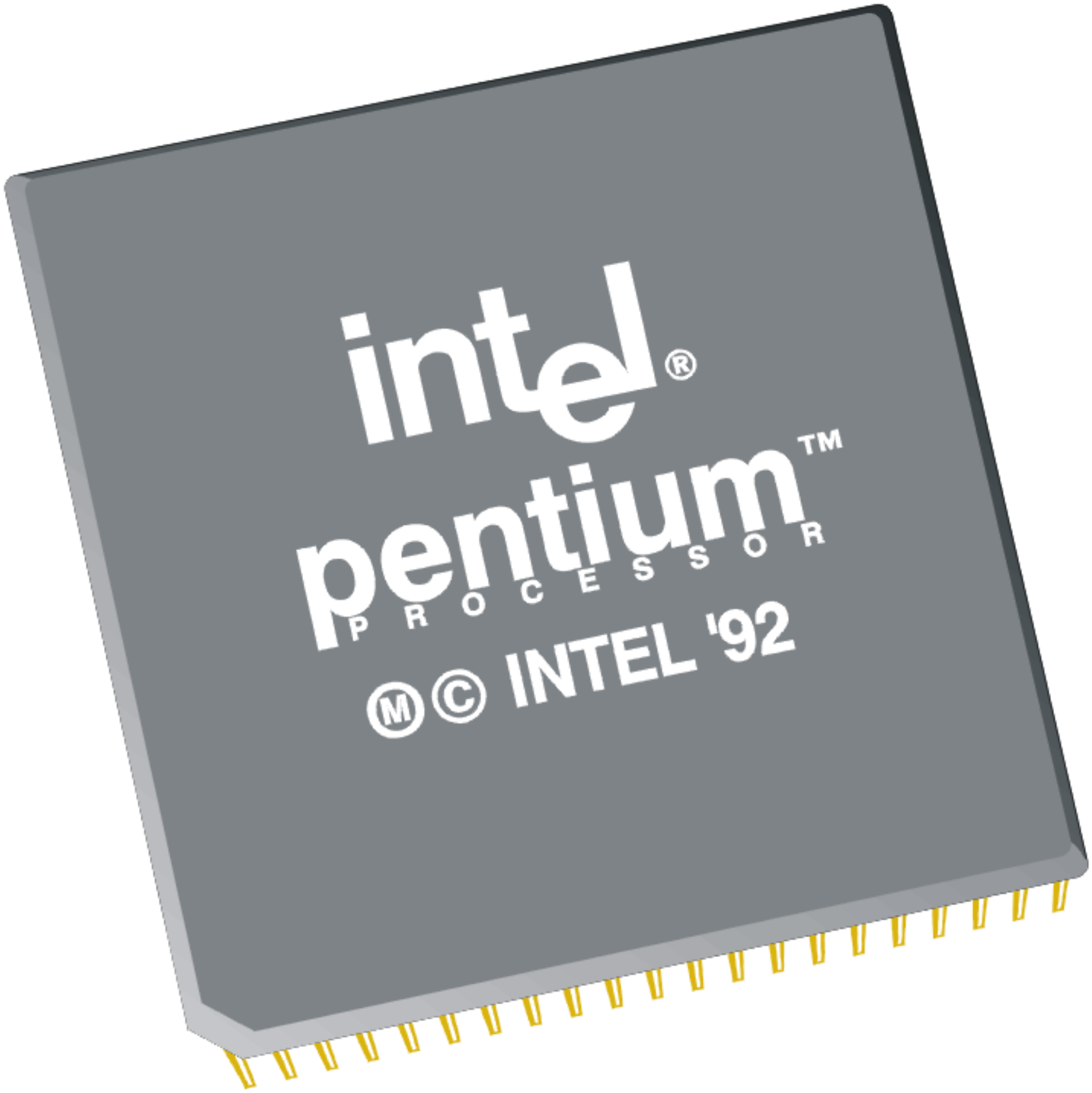





M'SIA 0027U
ULN2803A


silicon chip
19 mm
5 mm
notch
small dot
plastic case
0.1 inch


AMD
64
Athlon FX



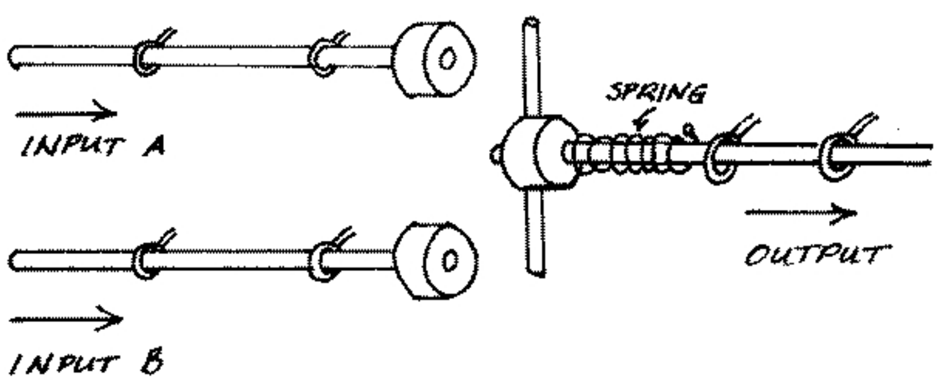


FIGURE 4

Mechanical implementation of the OR function

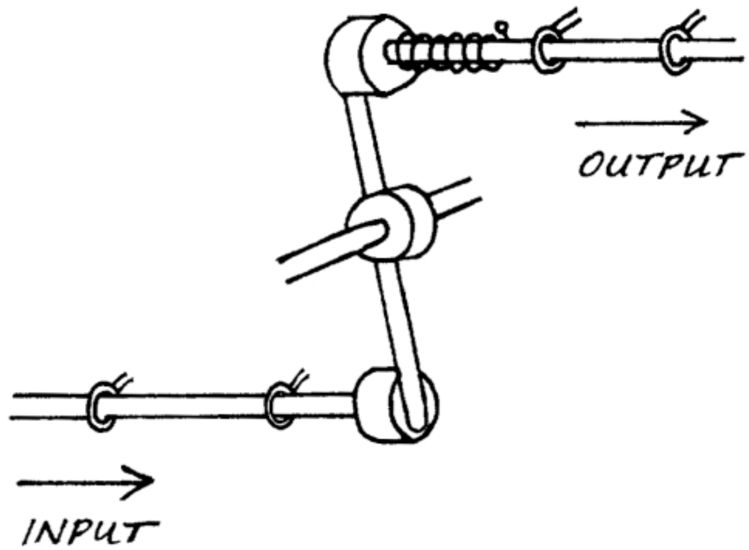


FIGURE 5

Mechanical inverter

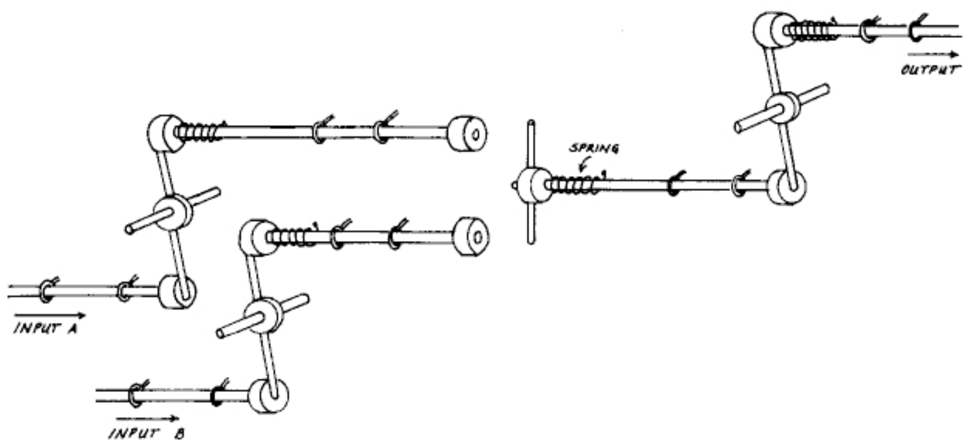


FIGURE 6

An And **block constructed by connecting an** Or **block to inverters**

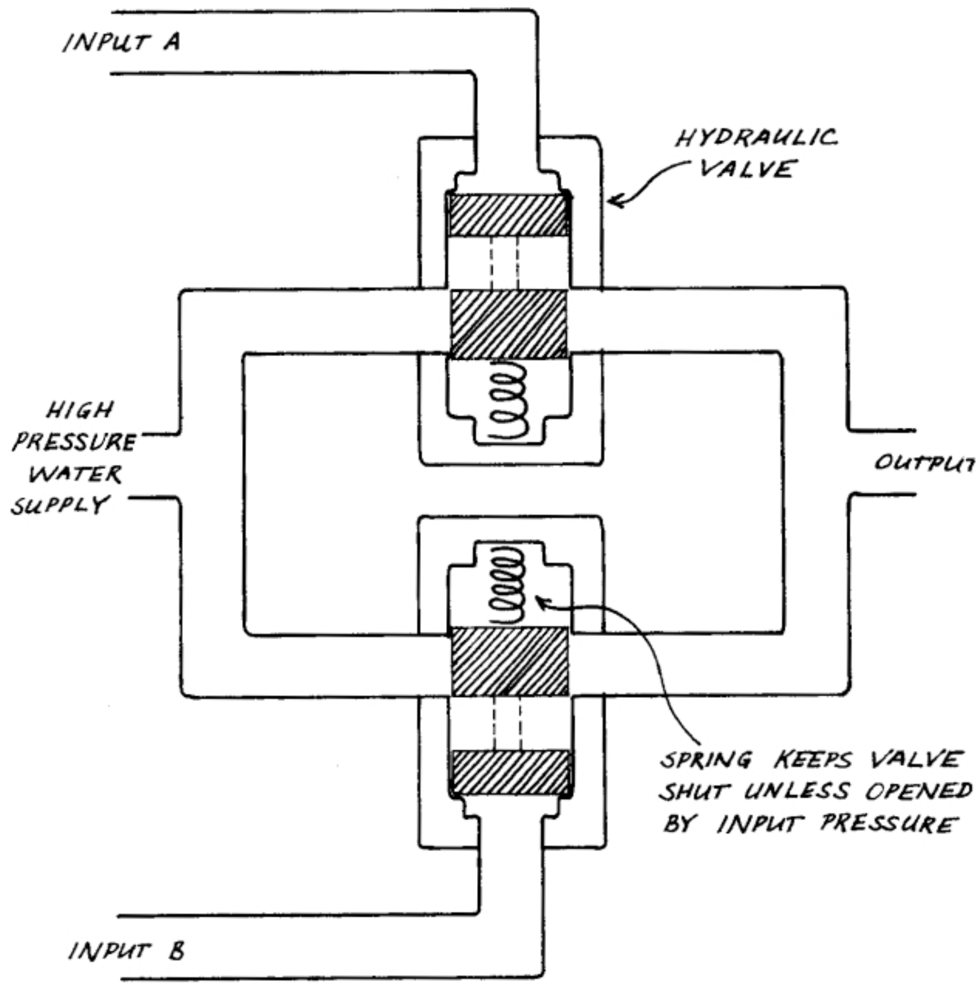
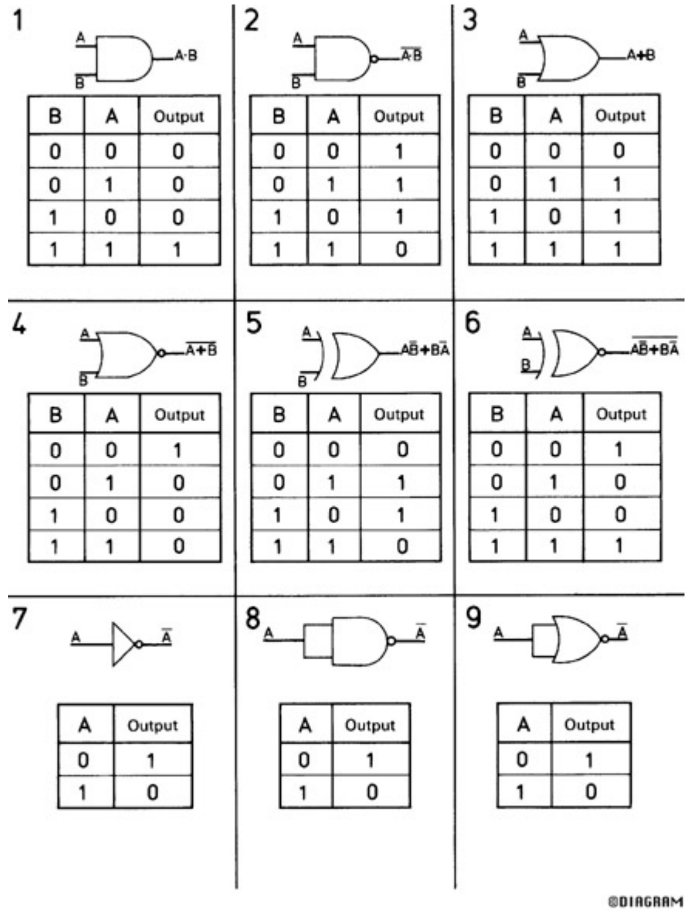


FIGURE 7

An Or **block built with hydraulic valves**



1

B	A	Output
0	0	0
0	1	0
1	0	0
1	1	1

2

B	A	Output
0	0	1
0	1	1
1	0	1
1	1	0

3

B	A	Output
0	0	0
0	1	1
1	0	1
1	1	1

4

B	A	Output
0	0	1
0	1	0
1	0	0
1	1	0

5

B	A	Output
0	0	0
0	1	1
1	0	1
1	1	0

6

B	A	Output
0	0	1
0	1	0
1	0	0
1	1	1

7

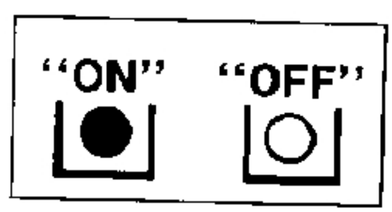
A	Output
0	1
1	0

8

A	Output
0	1
1	0

9

A	Output
0	1
1	0



No element in the description of physics shows itself as closer to primordial than the elementary quantum phenomenon, that is, the elementary device-intermediated act of posing a yes-no physical question and eliciting an answer or, in brief, the elementary act of observer-participancy. Otherwise stated, every physical quantity, every it, derives its ultimate significance from bits, binary yes-or-no indications, a conclusion which we epitomize in the phrase, *it from bit*.

— John Archibald Wheeler, "Information, physics, quantum: the search for links"

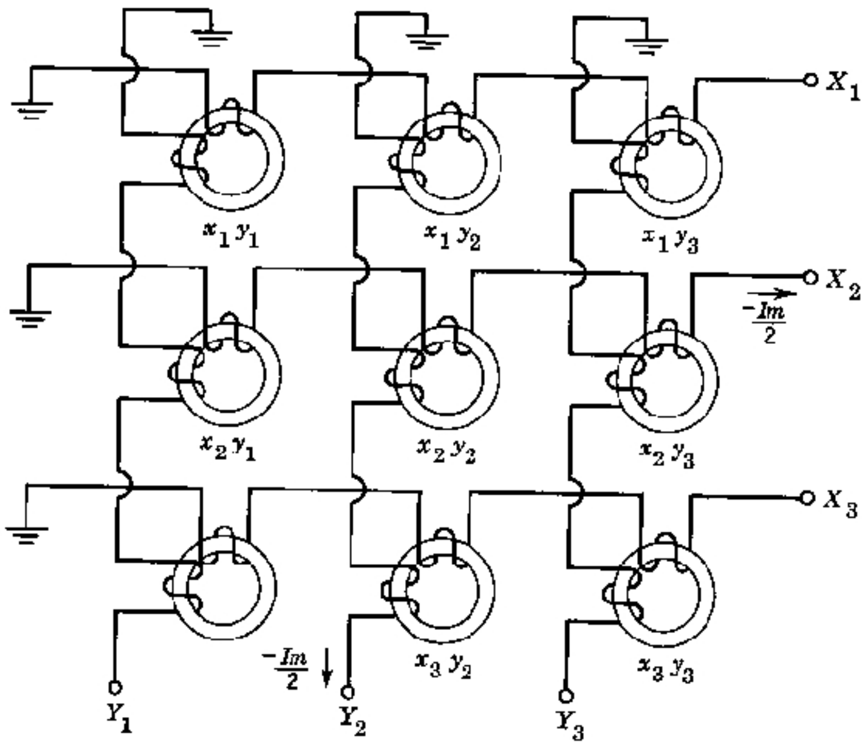
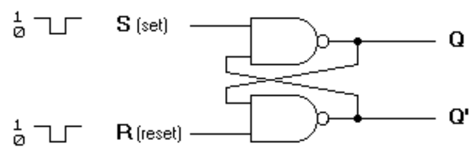


Fig. 7 · 4 Magnetic-core array.



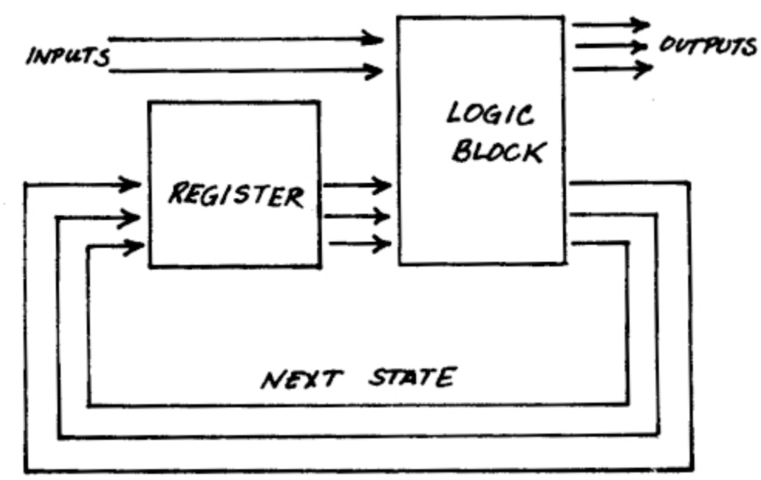


FIGURE 14

Finite-state machine, with logic block feeding register

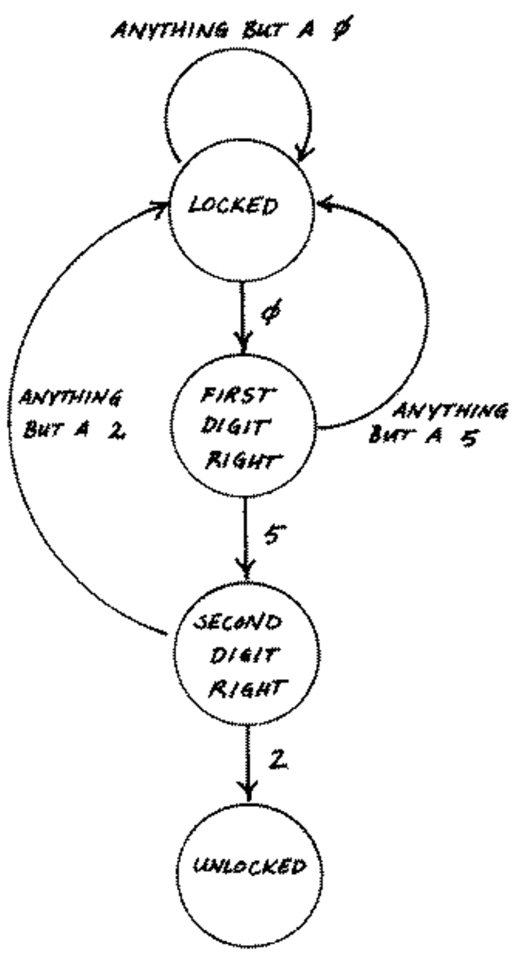
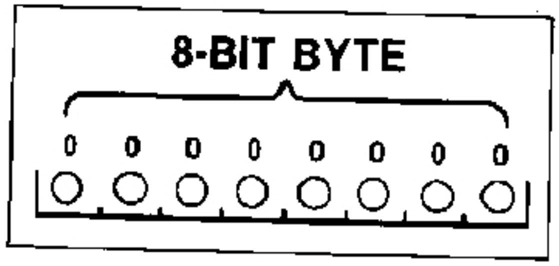


FIGURE 15

State diagram for a lock with combination 0-5-2



Character	6-Bit internal code	7-Bit ASCII code	8-Bit EBCDIC code	12-Bit card code
A	010 001	100 0001	1100 0001	12,1
B	010 010	100 0010	1100 0010	12,2
C	010 011	100 0011	1100 0011	12,3
D	010 100	100 0100	1100 0100	12,4
E	010 101	100 0101	1100 0101	12,5
F	010 110	100 0110	1100 0110	12,6
G	010 111	100 0111	1100 0111	12,7
H	011 000	100 1000	1100 1000	12,8
I	011 001	100 1001	1100 1001	12,9
J	100 001	100 1010	1101 0001	11,1
K	100 010	100 1011	1101 0010	11,2
L	100 011	100 1100	1101 0011	11,3
M	100 100	100 1101	1101 0100	11,4
N	100 101	100 1110	1101 0101	11,5
O	100 110	100 1111	1101 0110	11,6
P	100 111	101 0000	1101 0111	11,7
Q	101 000	101 0001	1101 1000	11,8
R	101 001	101 0010	1101 1001	11,9
S	110 010	101 0011	1110 0010	0,2
T	110 011	101 0100	1110 0011	0,3
U	110 100	101 0101	1110 0100	0,4
V	110 101	101 0110	1110 0101	0,5
W	110 110	101 0111	1110 0110	0,6
X	110 111	101 1000	1110 0111	0,7
Y	111 000	101 1001	1110 1000	0,8
Z	111 001	101 1010	1110 1001	0,9

BITS B7 B6 B5 / B4 B3 B2 B1				Ø Ø Ø	Ø Ø 1	Ø 1 Ø	Ø 1 1	1 Ø Ø	1 Ø 1	1 1 Ø	1 1 1
B4	B3	B2	B1	CONTROL		NUMBERS & SYMBOLS		UPPERCASE		LOWERCASE	
Ø	Ø	Ø	Ø	NUL 0	DLE 16	SP 32	0 48	@ 64	P 80	` 96	p 112
Ø	Ø	Ø	1	SOH 1	DC1 17	! 33	1 49	A 65	Q 81	a 97	q 113
Ø	Ø	1	Ø	STX 2	DC2 18	" 34	2 50	B 66	R 82	b 98	r 114
Ø	Ø	1	1	ETX 3	DC3 19	# 35	3 51	C 67	S 83	c 99	s 115
Ø	1	Ø	Ø	EOT 4	DC4 20	$ 36	4 52	D 68	T 84	d 100	t 116
Ø	1	Ø	1	ENQ 5	NAK 21	% 37	5 53	E 69	U 85	e 101	u 117
Ø	1	1	Ø	ACK 6	SYN 22	& 38	6 54	F 70	V 86	f 102	v 118
Ø	1	1	1	BEL 7	ETB 23	' 39	7 55	G 71	W 87	g 103	w 119
1	Ø	Ø	Ø	BS 8	CAN 24	(40	8 56	H 72	X 88	h 104	x 120
1	Ø	Ø	1	HT 9	EM 25	) 41	9 57	I 73	Y 89	i 105	y 121
1	Ø	1	Ø	LF 10	SUB 26	* 42	: 58	J 74	Z 90	j 106	z 122
1	Ø	1	1	VT 11	ESC 27	+ 43	; 59	K 75	[91	k 107	{ 123
1	1	Ø	Ø	FF 12	FS 28	, 44	< 60	L 76	\ 92	l 108	| 124
1	1	Ø	1	CR 13	GS 29	– 45	= 61	M 77	] 93	m 109	} 125
1	1	1	Ø	SO 14	RS 30	. 46	> 62	N 78	↑ 94	n 110	~ 126
1	1	1	1	SI 15	US 31	/ 47	? 63	O 79	_ 95	o 111	↓ 127

Figure 1-8 The message "Hi Sue." in ASCII

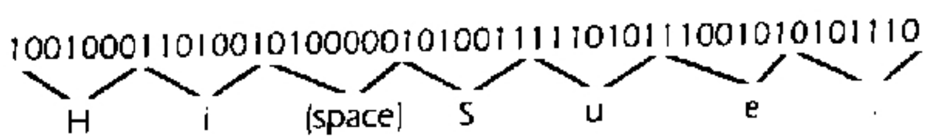
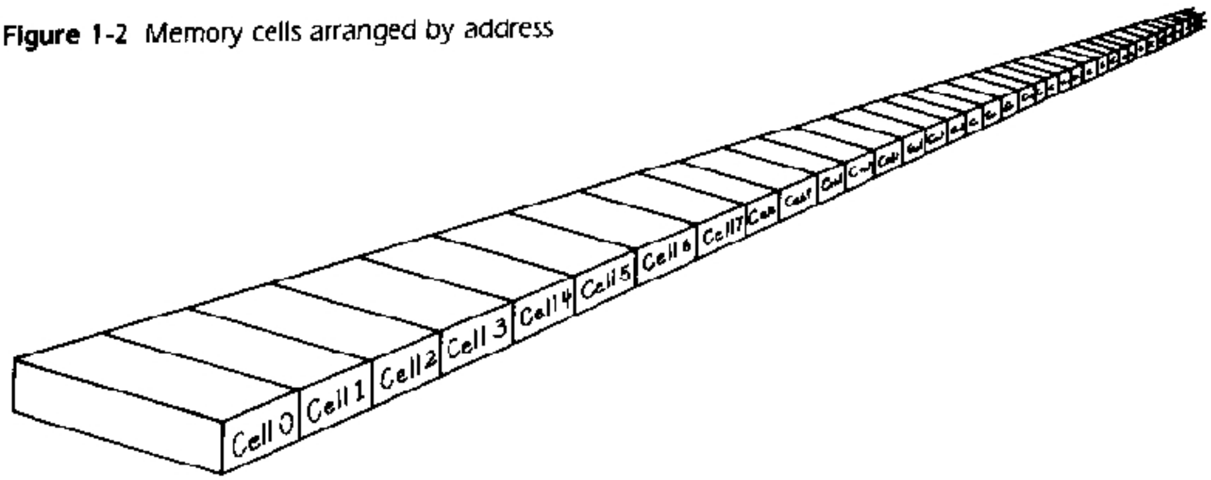
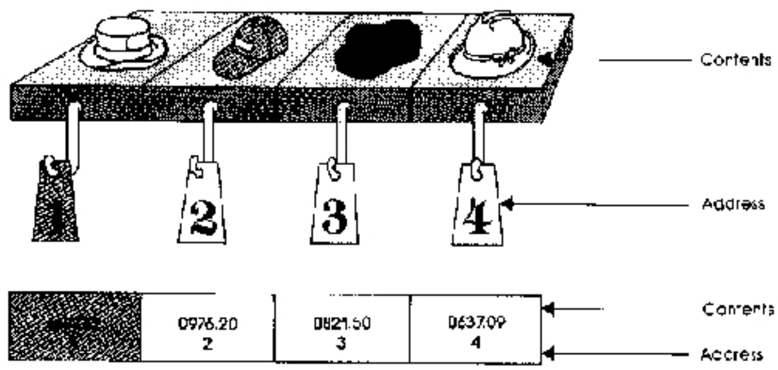
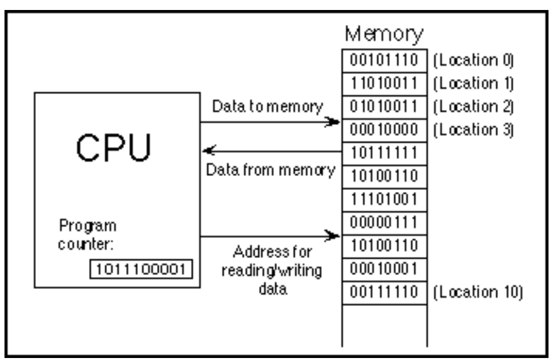


Figure 1-2 Memory cells arranged by address







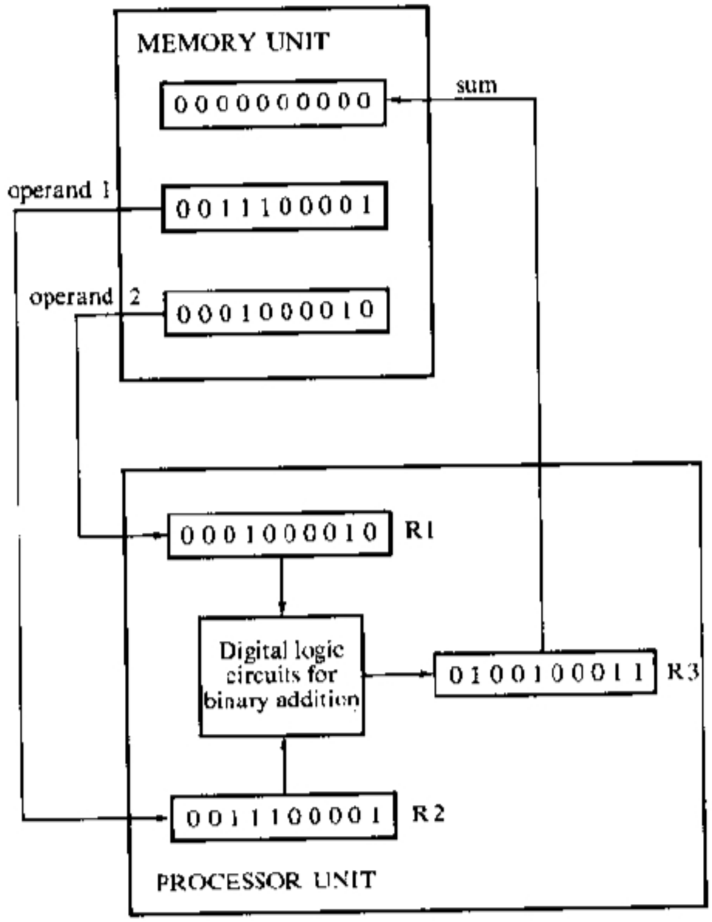
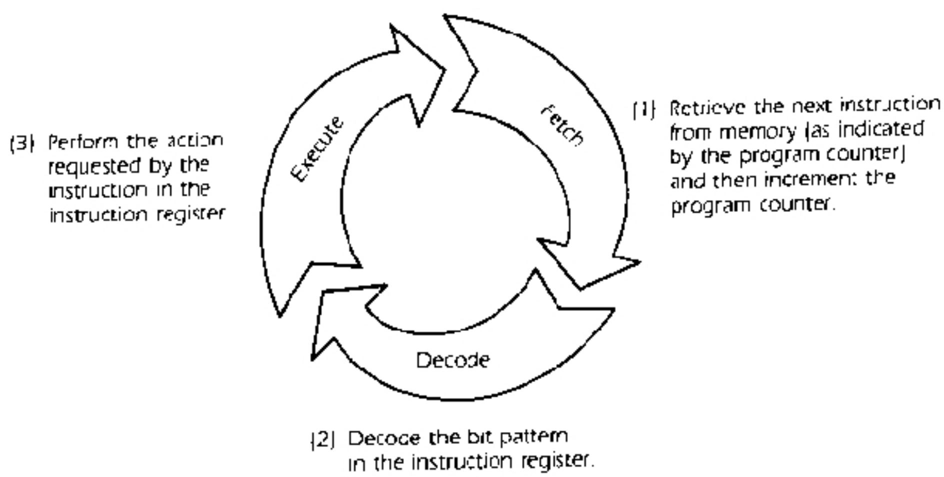


Figure 1-3 Example of binary information processing

Figure 2-6 The machine cycle



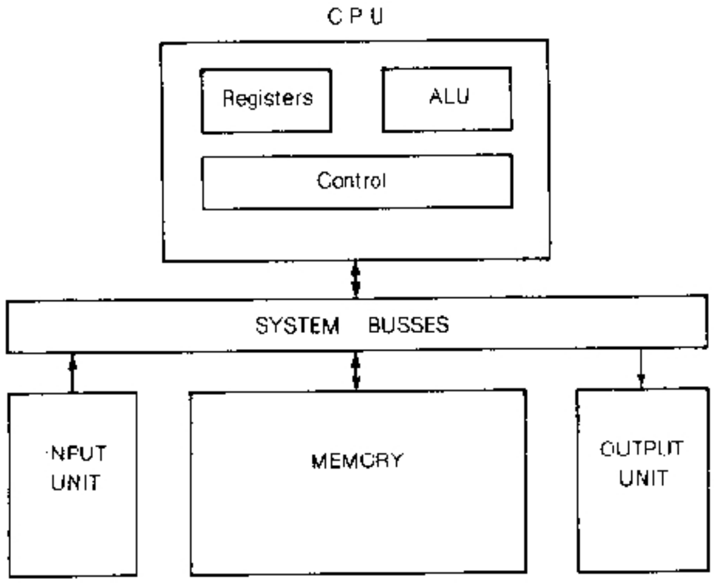
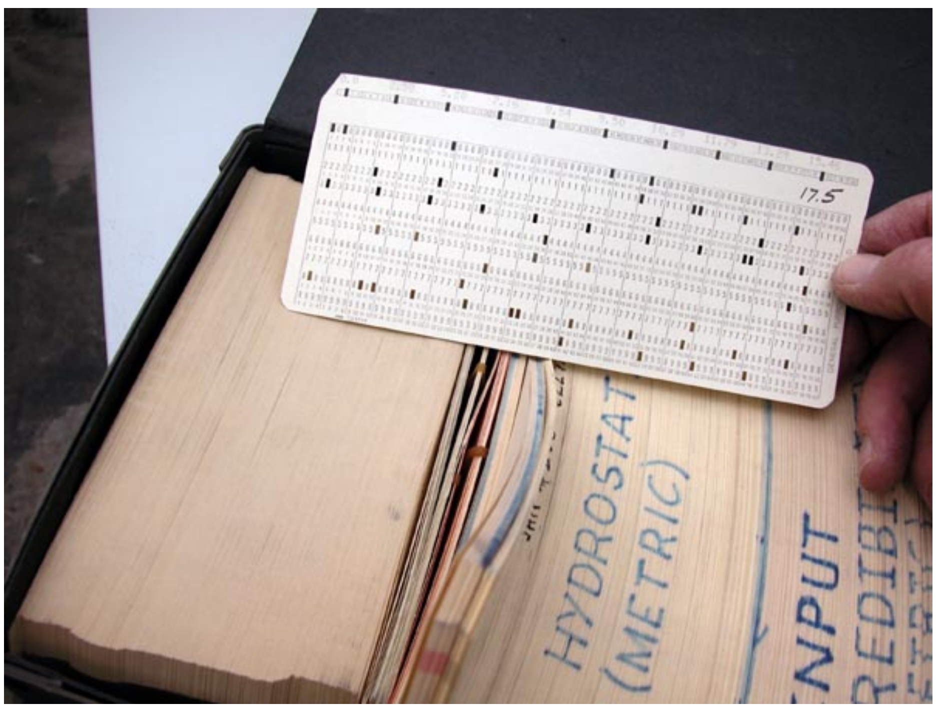


Figure 2-1. Main Units of a Computer.



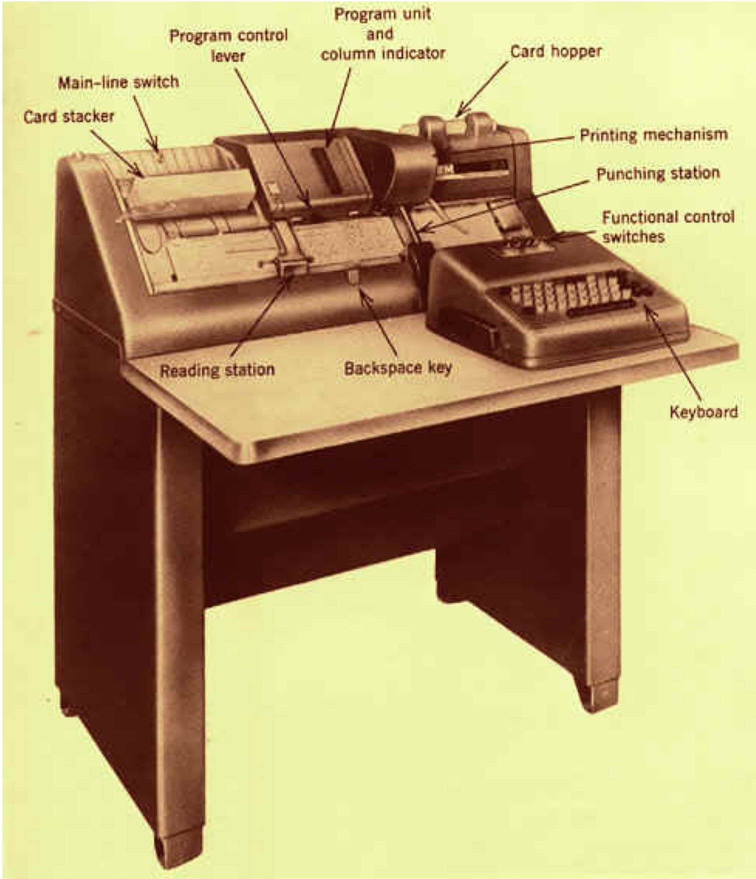


FIGURE 4.11 IBM 026 Keypunch (Courtesy IBM).

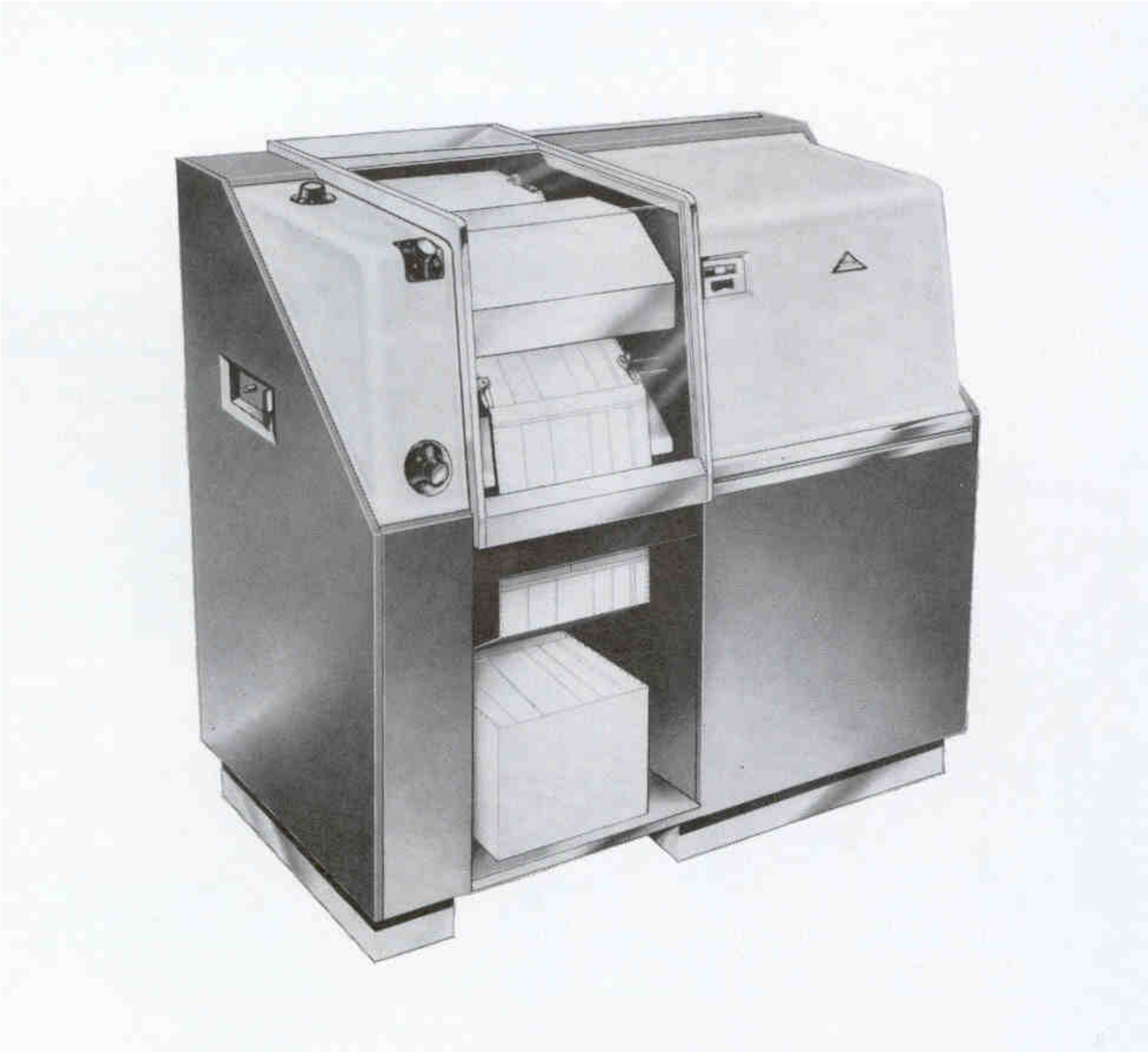
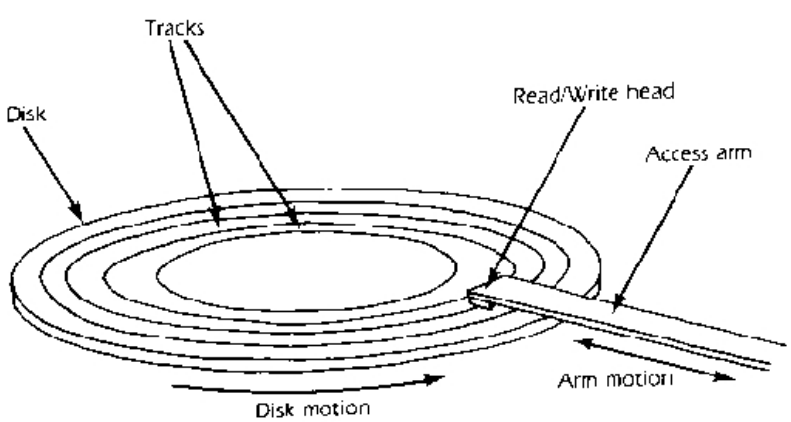
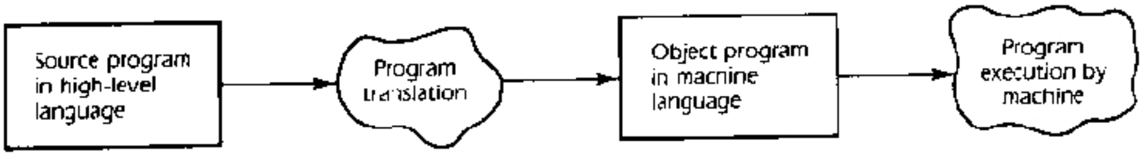


Figure 1-7 A disk storage system





(b) Multistep translate/execute sequence required by the translation process

---

re 4.2 ***Program to read two integers and print their sum.***

```
RDPRT    CSECT
*
*  THIS PROGRAM READS TWO NONNEGATIVE INTEGERS FROM A DA
*   THAT IS IN THE INPUT DATA STREAM, COMPUTES THEIR SUM
*   PRINTS THE RESULT IN A SINGLE LINE ON THE PRINTER.
*
*  THE INPUT DATA IS ASSUMED TO BE REPRESENTED AS 10-BYT
*   DECIMAL INTEGERS, WITH EITHER LEADING ZEROES OR LEAD
*
*  THE SUM IS PRINTED AS A 10-BYTE UNSIGNED DECIMAL INTE
*   LEADING ZEROES PRINTED AS ZEROES.
*
         STM   14,12,12(13)   STANDARD ENTRY
         BALR  12,0
         USING *,12
         ST    13,SAVE+4
         LA    13,SAVE
*
*  OPEN  INPUT AND OUTPUT FILES
         OPEN  (TRANS,(INPUT))         CARDS FROM INPUT DA
         OPEN  (PRINTER,(OUTPUT))      PRINTER IS SYSOUT
*
*  READ ONE DATA CARD
         GET   TRANS,CARD              INPUT AREA IS 'CARD'
*
*  CONVERT EACH NUMBER TO 32-BIT BINARY
         PACK  BCD,INT1                FIRST INTEGER IS AT
         CVB   5,BCD                   ITS 32-BIT BINARY IN
         ST    5,X                      STORED AT 'X'.
         PACK  BCD,INT2                SECOND INTEGER IS AT
         CVB   5,BCD                   ITS 32-BIT BINARY IN
         ST    5,Y                      STORED AT 'Y'.
*
*  COMPUTE THE SUM, AND STORE BINARY INTEGER RESULT AT 'Z
         L     3,X              COMPUTE THE SUM OF THE INTE
         A     3,Y              AT SYMBOLIC ADDRESSES X & Y
         ST    3,Z              STORE THE SUM AT SYMBOLIC A
*
```

FORTRAN Arithmetic Statement	Equivalent Algebraic Equation
`Y = (X + 6.)/B - C`	$y = \frac{x + 6}{b} - c$
`Y = X + 6./(B - C)`	$y = x + \frac{6}{b - c}$
`Y = X + 6./B - C`	$y = x + \frac{6}{b} - c$

PECANS TOASTED WITH CHIVES

4 *cups shelled pecan halves*
4 *beef bouillon cubes, crumbled*
½ *cup dried chopped chives*
¼ *cup (½ stick) butter*
Salt and freshly ground pepper, if desired

1. Preheat oven to 300°.
2. Toast pecans in shallow pan in oven for 30 minutes, stirring occasionally.
3. Add crumbled bouillon cubes, chives, and butter, mixing well.
4. Return to oven for 15 minutes and continue to toast, again stirring often.
5. Add salt and pepper, if desired.
6. The nuts may be served immediately, warm or cool, or stored in an airtight container for future use. They will keep for a month in the cupboard, or indefinitely if frozen.

Makes about 1 quart.

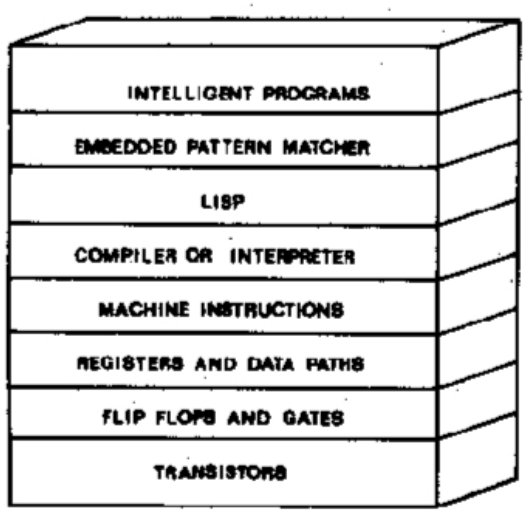
---

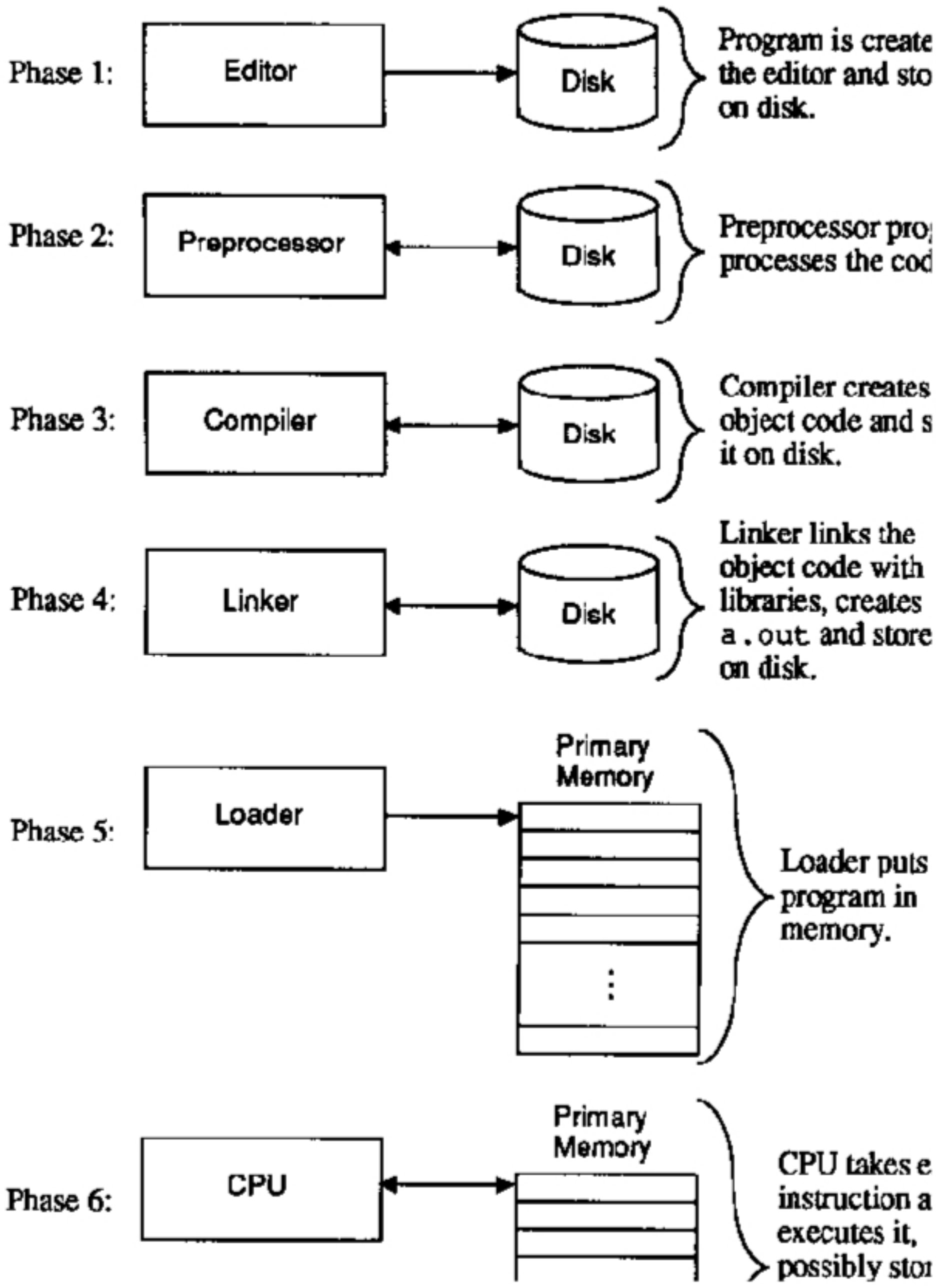
begin
1. Request the list of names and call it **NameList**.
2. Request the name being sought and call it **KeyName**;
3. On a black board called **Count** write the number zero;
4. Repeat the following for each name on **NameList**:
   if the name on **NameList** is the same as **KeyName**
   then add one to the number written on **Count**;
   {the old number is erased, leaving only one number on **Count**}
5. Announce that the desired answer is written on **Count**

end.

```
on mouseUp
  set the name of me to "Click to Stop!"
  repeat until (the mouse = down)
    put "play " & (card field Sound) & ¬
    " tempo 400" && randomnote() into str
    wait until the sound is "done"
    do str
  end repeat
  set the name of me to "Random Notes"
end mouseUp
```

```
(defoperator do-laundry
  :level :causal
  :for (achieving (clean :?))
  :filters ((believed `(clothing ,?1)))
  :steps ((TAKE-TO-MACHINE
           (achieve `(on ,?1 laundry-machine)
                    :call-back
                    #'(lambda ()
                        (finish TAKE-TO-MACHINE))))
          (TAKE-TO-BED-WHEN-DONE
           (achieve `(after laundry-done on ,?1 bed)
                    :call-back
                    #'(lambda () (succeed)))))
  :initially (TAKE-TO-MACHINE)
  :transitions ((TAKE-TO-MACHINE :>
                   TAKE-TO-BED-WHEN-DONE))
  :on-success ((believe `(clean ,?1))))
```




Phase 1:
Editor
Disk
Program is create
the editor and sto
on disk.
Phase 2:
Preprocessor
Disk
Preprocessor pro
processes the cod
Phase 3:
Compiler
Disk
Compiler creates
object code and s
it on disk.
Phase 4:
Linker
Disk
Linker links the
object code with
libraries, creates
a.out and store
on disk.
Phase 5:
Loader
Primary
Memory
Loader puts
program in
memory.
Phase 6:
CPU
Primary
Memory
CPU takes e
instruction a
executes it,
possibly stor