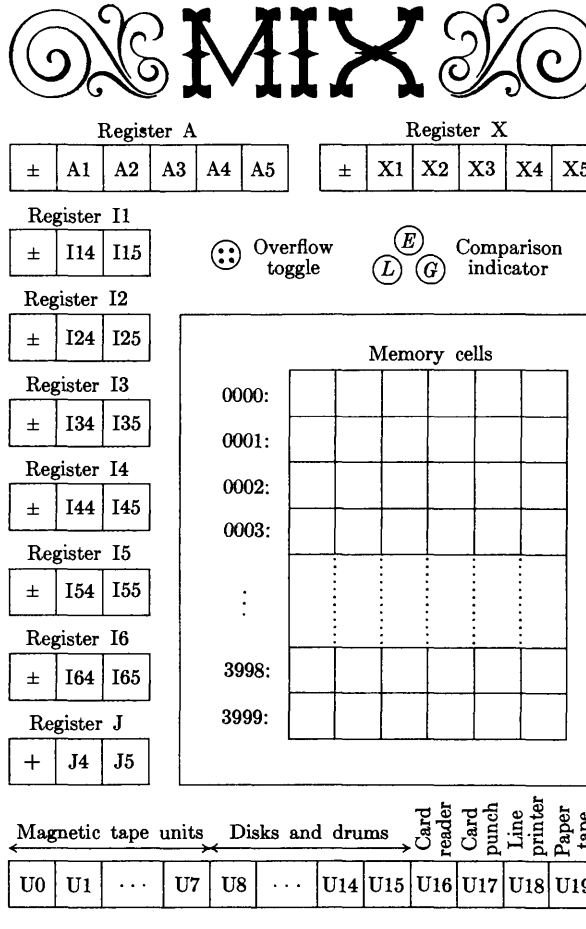

Fig. 13. The MIX computer.

Besides its registers, MIX contains

an *overflow toggle* (a single bit that is either "on" or "off");
a *comparison indicator* (having three values: LESS, EQUAL, or GREATER);
memory (4000 words of storage, each word with five bytes and a sign);
and *input-output devices* (cards, tapes, disks, etc.).

Partial fields of words. The five bytes and sign of a computer word are numbered as follows:

0	1	2	3	4	5
±	Byte	Byte	Byte	Byte	Byte

. (2)

Table 1

Character code:

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
␣	A	B	C	D	E	F	G	H	I	Δ	J	K	L	M	N	O	P	Q	R	Σ	Π	S	T	U

00	*1*	**01**	*2*	**02**	*2*	**03**	*10*

No operation
`NOP(0)`
		rA ← rA + V
		`ADD(0:5)`
		`FADD(6)`
				rA ← rA − V
				`SUB(0:5)`
				`FSUB(6)`
						rAX ← rA × V
						`MUL(0:5)`
						`FMUL(6)`	

08	*2*	**09**	*2*	**10**	*2*	**11**	*2*
rA ← V `LDA(0:5)`		rI1 ← V `LD1(0:5)`		rI2 ← V `LD2(0:5)`		rI3 ← V `LD3(0:5)`	
16	*2*	**17**	*2*	**18**	*2*	**19**	*2*
rA ← −V `LDAN(0:5)`		rI1 ← −V `LD1N(0:5)`		rI2 ← −V `LD2N(0:5)`		rI3 ← −V `LD3N(0:5)`	
24	*2*	**25**	*2*	**26**	*2*	**27**	*2*
M(F) ← rA `STA(0:5)`		M(F) ← rI1 `ST1(0:5)`		M(F) ← rI2 `ST2(0:5)`		M(F) ← rI3 `ST3(0:5)`	
32	*2*	**33**	*2*	**34**	*1*	**35**	*1 + T*
M(F) ← rJ `STJ(0:2)`		M(F) ← 0 `STZ(0:5)`		Unit F busy? `JBUS(0)`		Control, unit F `IOC(0)`	
40	*1*	**41**	*1*	**42**	*1*	**43**	*1*
rA : 0, jump `JA[+]`		rI1 : 0, jump `J1[+]`		rI2 : 0, jump `J2[+]`		rI3 : 0, jump `J3[+]`	
48	*1*	**49**	*1*	**50**	*1*	**51**	*1*

rA ← [rA]? ± M
`INCA(0) DECA(1)`
`ENTA(2) ENNA(3)`
		rI1 ← [rI1]? ± M
		`INC1(0) DEC1(1)`
		`ENT1(2) ENN1(3)`
				rI2 ← [rI2]? ± M
				`INC2(0) DEC2(1)`
				`ENT2(2) ENN2(3)`
						rI3 ← [rI3]? ± M
						`INC3(0) DEC3(1)`
						`ENT3(2) ENN3(3)`	

56	*2*	**57**	*2*	**58**	*2*	**59**	*2*

CI ← rA(F) : V
`CMPA(0:5)`
`FCMP(6)`
		CI ← rI1(F) : V
		`CMP1(0:5)`
				CI ← rI2(F) : V
				`CMP2(0:5)`
						CI ← rI3(F) : V
						`CMP3(0:5)`	

General form:

C	*t*
Description `OP(F)`	

C = operation code, (5:5) field of instruction
F = op variant, (4:4) field of instruction
M = address of instruction after indexing
V = M(F) = contents of F field of location M
`OP` = symbolic name for operation
`(F)` = normal F setting
t = execution time; T = interlock time

25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55
V	W	X	Y	Z	0	1	2	3	4	5	6	7	8	9	.	,	(	)	+	−	*	/	=	$	<	>	@	;	:	'

04	*12*	**05**	*10*	**06**	*2*	**07**	*1 + 2F*
rA ← rAX/V rX ← remainder DIV(0:5) FDIV(6)		Special NUM(0) CHAR(1) HLT(2)		Shift M bytes SLA(0) SRA(1) SLAX(2) SRAX(3) SLC(4) SRC(5)		Move F words from M to rI1 MOVE(1)	
12	*2*	**13**	*2*	**14**	*2*	**15**	*2*
rI4 ← V LD4(0:5)		rI5 ← V LD5(0:5)		rI6 ← V LD6(0:5)		rX ← V LDX(0:5)	
20	*2*	**21**	*2*	**22**	*2*	**23**	*2*
rI4 ← −V LD4N(0:5)		rI5 ← −V LD5N(0:5)		rI6 ← −V LD6N(0:5)		rX ← −V LDXN(0:5)	
28	*2*	**29**	*2*	**30**	*2*	**31**	*2*
M(F) ← rI4 ST4(0:5)		M(F) ← rI5 ST5(0:5)		M(F) ← rI6 ST6(0:5)		M(F) ← rX STX(0:5)	
36	*1 + T*	**37**	*1 + T*	**38**	*1*	**39**	*1*
Input, unit F IN(0)		Output, unit F OUT(0)		Unit F ready? JRED(0)		Jumps JMP(0) JSJ(1) JOV(2) JNOV(3) also [*] below	
44	*1*	**45**	*1*	**46**	*1*	**47**	*1*
rI4 : 0, jump J4[+]		rI5 : 0, jump J5[+]		rI6 : 0, jump J6[+]		rX : 0, jump JX[+]	
52	*1*	**53**	*1*	**54**	*1*	**55**	*1*
rI4 ← [rI4]? ± M INC4(0) DEC4(1) ENT4(2) ENN4(3)		rI5 ← [rI5]? ± M INC5(0) DEC5(1) ENT5(2) ENN5(3)		rI6 ← [rI6]? ± M INC6(0) DEC6(1) ENT6(2) ENN6(3)		rX ← [rX]? ± M INCX(0) DECX(1) ENTX(2) ENNX(3)	
60	*2*	**61**	*2*	**62**	*2*	**63**	*2*
CI ← rI4(F) : V CMP4(0:5)		CI ← rI5(F) : V CMP5(0:5)		CI ← rI6(F) : V CMP6(0:5)		CI ← rX(F) : V CMPX(0:5)	

rA = register A
rX = register X
rAX = registers A and X as one
rIi = index register i, $1 \le i \le 6$
rJ = register J
CI = comparison indicator

[*]:		[+]:
JL(4)	$<$	N(0)
JE(5)	$=$	Z(1)
JG(6)	$>$	P(2)
JGE(7)	$\ge$	NN(3)
JNE(8)	$\neq$	NZ(4)
JLE(9)	$\le$	NP(5)