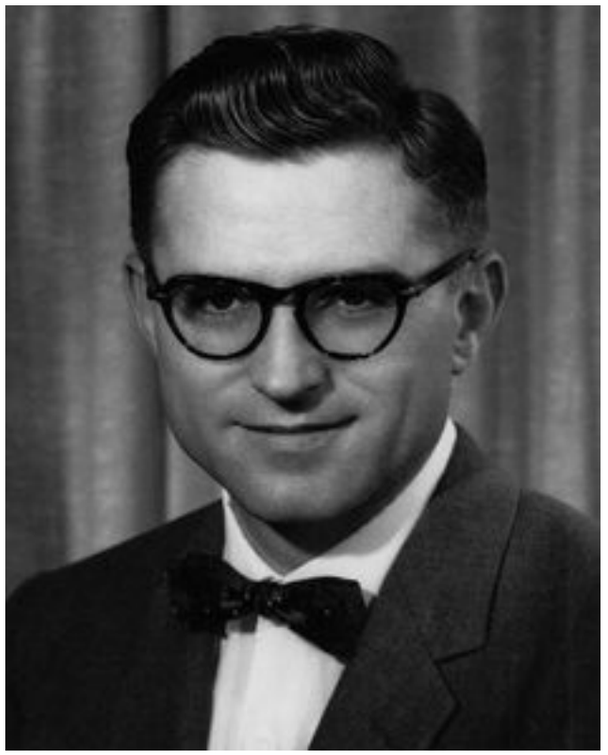
IBM 360

& Tomasulo's Algorithm



Amdahl

Amdahl's parallelism law: If a computation has a serial component S and a parallel component P, then the maximum speedup is (S+P)/S.

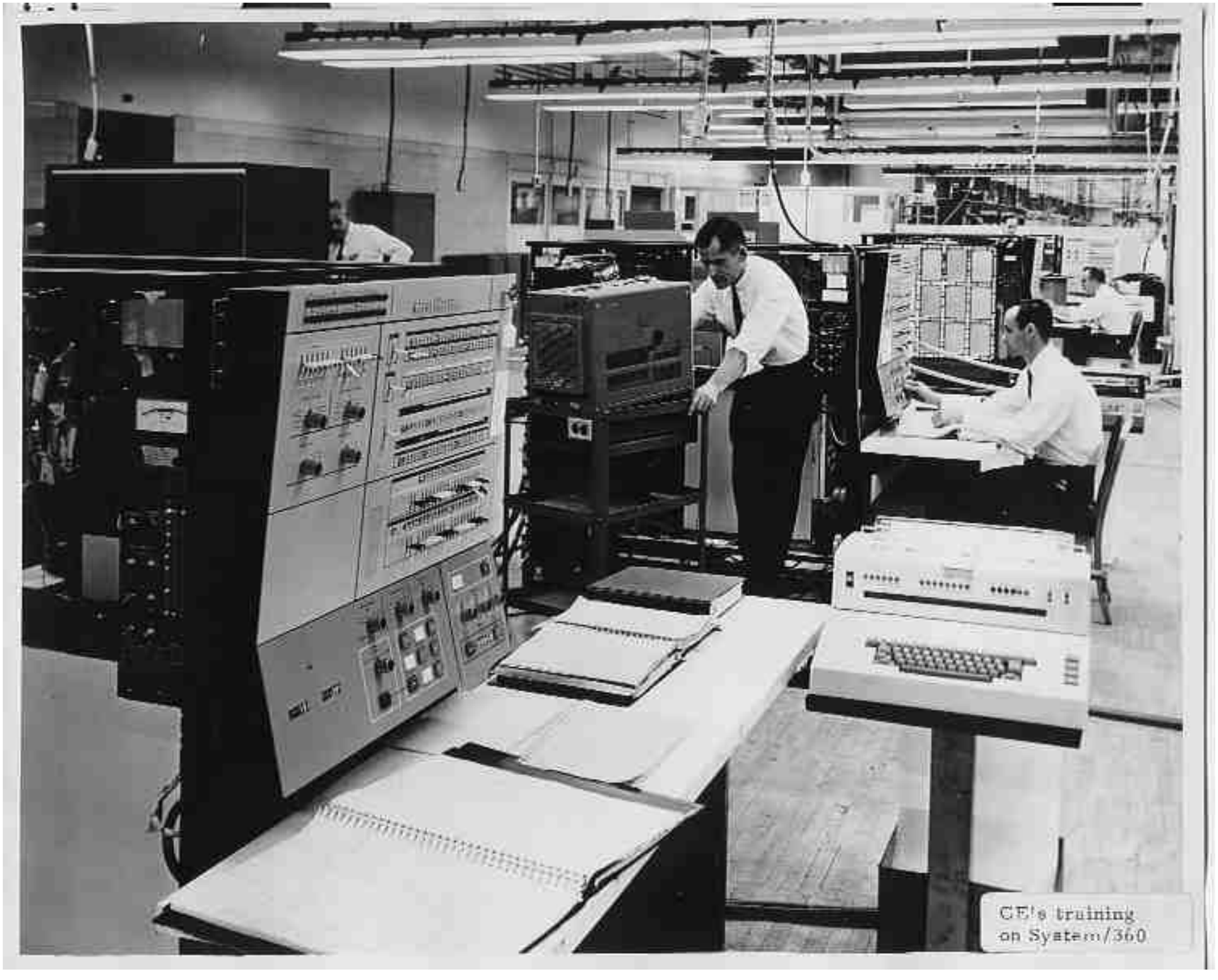
Amdahl's balanced system law: A system needs a bit of IO per second for each instruction per second: about 8 MIPS per MBps.

Amdahl's memory law: In a balanced system the MB/MIPS ratio is 1.

Amdahl's IO law: Programs do one IO per 50,000 instructions.


IBM
SYSTEM 360


IBM
360



CE's training on System/360

A — DISPLAY

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18

B — DISPLAY

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18

COMPARE B DATA FOR SYNC OR HOLD

HOLD
OFF

C — DISPLAY

1 2 3 4 5 6 7 8 9

ANY ERROR

SAMPLE INTLK

DISPLAY HOLD

ON
HOLD
OFF

COMPARE C DATA FOR GATE OR SAMPLE

SET CONTROL

TYPE

GATES

C COMP PROBE

6 IN

OFF

-ON

DISPLAY

HOLD CONTROL

B COMPARE

HOLD

OFF

ANY ERROR

HOLD

RESET

AUTO

END OP I (A)

OFF

MANUAL

SIGNAL LINES TO DEVICE ADAPTERS

ON = PLUS LEVEL

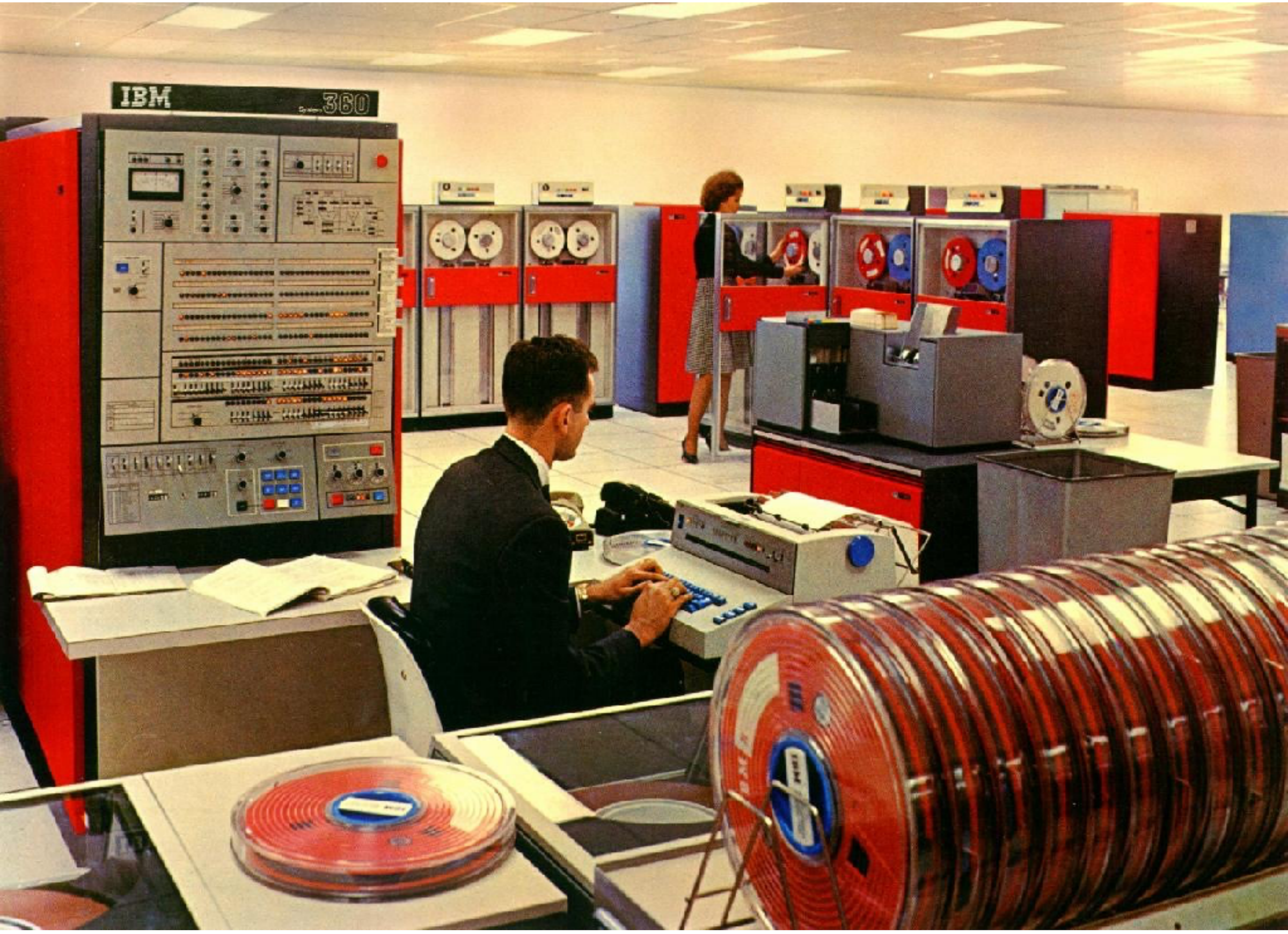
1 2 3 4

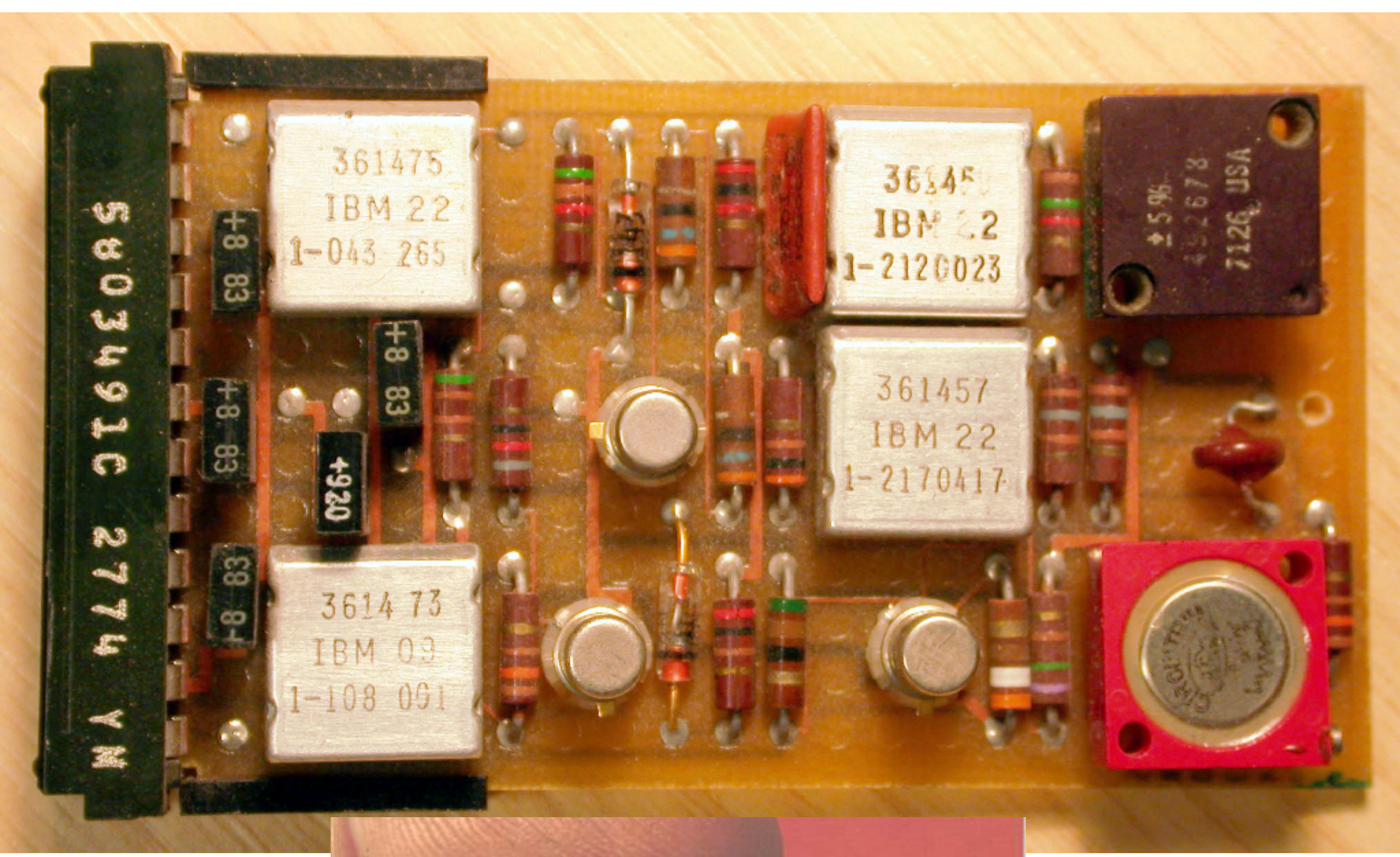
ON

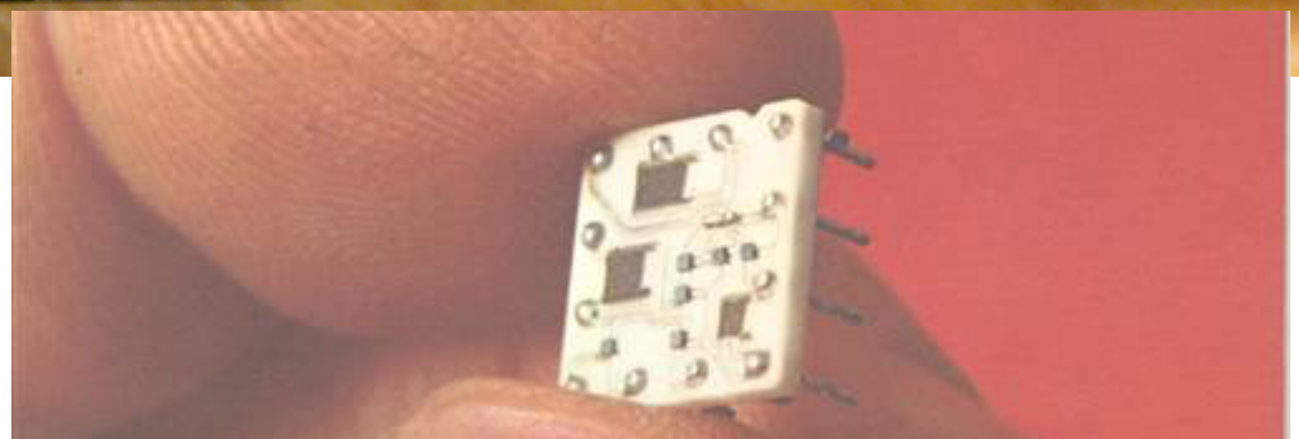
OFF

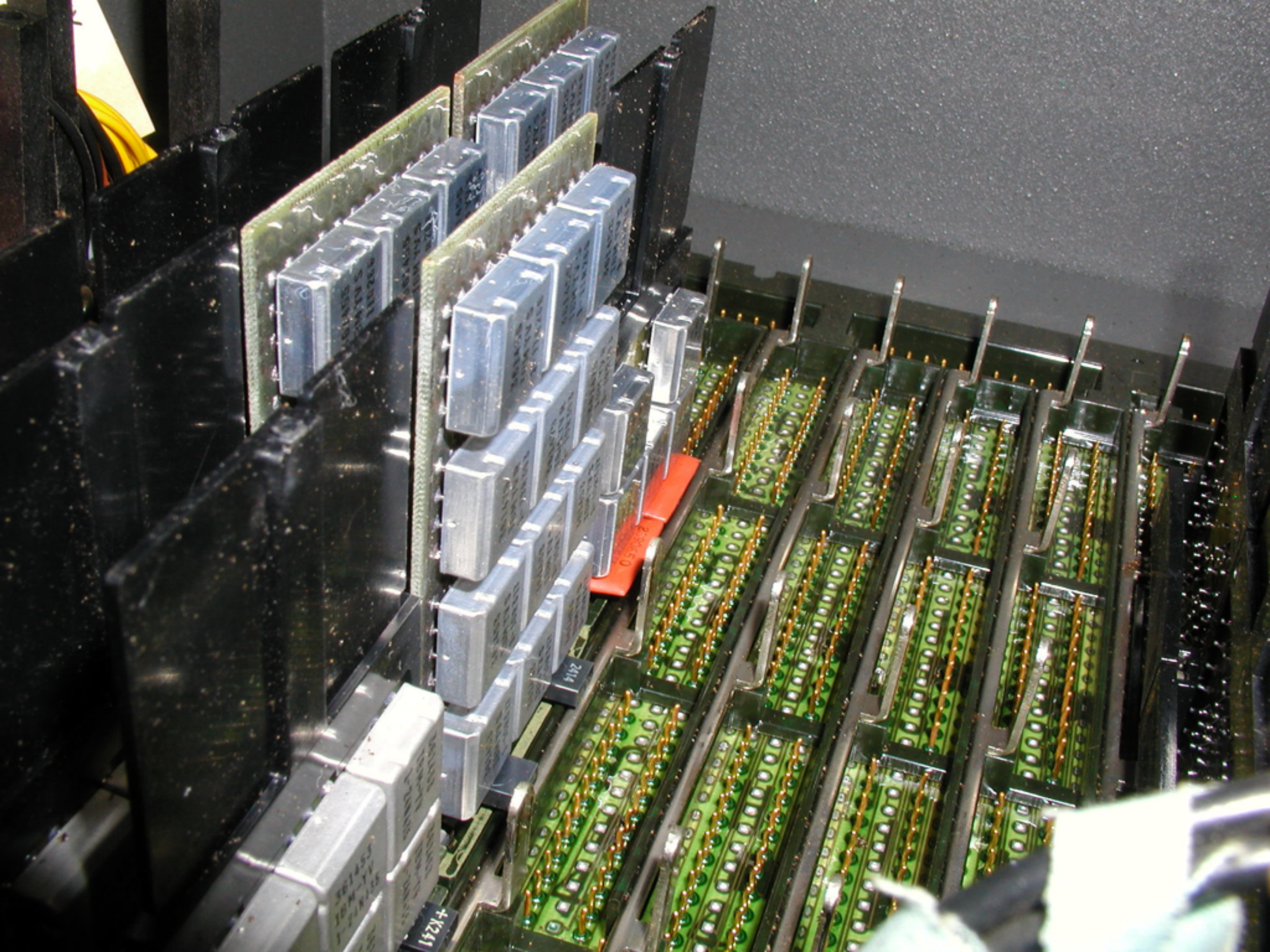
PULSED

5 RESET


IBM
360


361475
IBM 22
1-043 265
361473
IBM 03
1-108 091
361457
IBM 22
1-2170417
+8 83
+920




+K241
Z414

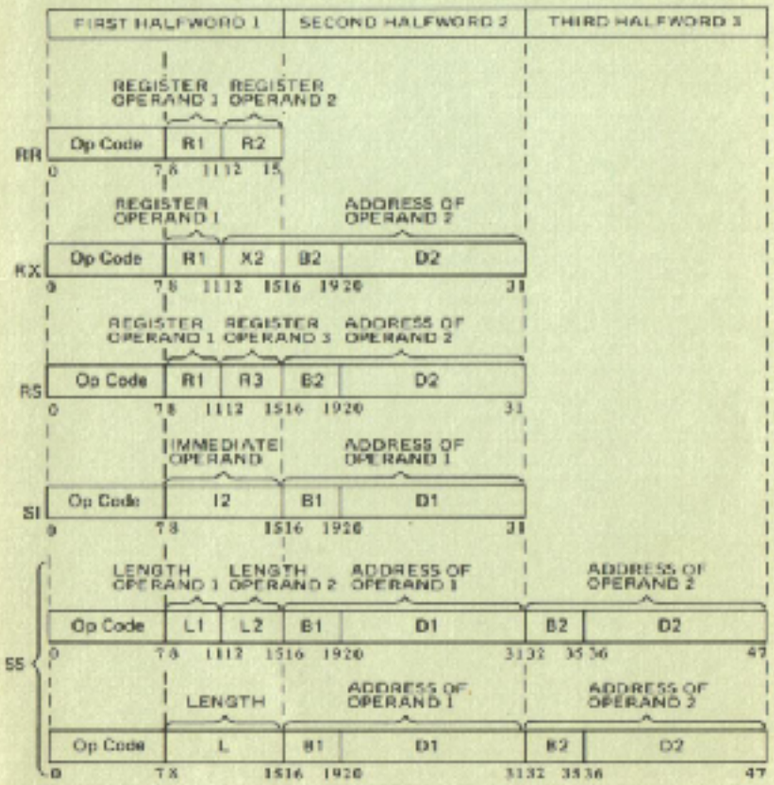

IBM
IBMシステム／360 本邦第1号機到着
A

OR (c)	OC	D6	SS	D1(L,B1),D2(B2)
Pack	PACK	F2	SS	D1(L1,B1),D2(L2,B2)
Read Direct (b,p)	RDD	85	SI	D1(B1),I2
Set Program Mask (n)	SPM	04	RR	R1
Set Storage Key (a,p)	SSK	08	RR	R1,R2
Set System Mask (p)	SSM	80	SI	D1(B1)
Shift Left Double (c)	SLDA	8F	RS	R1,D2(B2)
Shift Left Double Logical	SLDL	8D	RS	R1,D2(B2)
Shift Left Single (c)	SLA	8B	RS	R1,D2(B2)
Shift Left Single Logical	SLL	89	RS	R1,D2(B2)
Shift Right Double (c)	SRDA	8E	RS	R1,D2(B2)
Shift Right Double Logical	SRDL	8C	RS	R1,D2(B2)
Shift Right Single (c)	SRA	8A	RS	R1,D2(B2)
Shift Right Single Logical	SRL	88	RS	R1,D2(B2)
Start I/O (c,p)	SIO	9C	SI	D1(B1)
Store	ST	50	RX	R1,D2(X2,B2)
Store Character	STC	42	RX	R1,D2(X2,B2)
Store Halfword	STH	40	RX	R1,D2(X2,B2)
Store Multiple	STM	90	RS	R1,R3,D2(B2)
Store Multiple Control (e,p)	STMC	B0	RS	R1,R3,D2(B2)
Subtract (c)	SR	1B	RR	R1,R2
Subtract (c)	S	5B	RX	R1,D2(X2,B2)
Subtract Decimal (c,d)	SP	FB	SS	D1(L1,B1),D2(L2,B2)
Subtract Halfword (c)	SH	4B	RX	R1,D2(X2,B2)
Subtract Logical (c)	SLR	1F	RR	R1,R2
Subtract Logical (c)	SL	5F	RX	R1,D2(X2,B2)
Supervisor Call	SVC	0A	RR	I
Test and Set (c)	TS	93	SI	D1(B1)
Test Channel (c,p)	TCH	9F	SI	D1(B1)
Test I/O (c,p)	TIO	9D	SI	D1(B1)
Test under Mask (c)	TM	91	SI	D1(B1),I2
Translate	TR	DC	SS	D1(L,B1),D2(B2)
Translate and Test (c)	TRT	DD	SS	D1(L,B1),D2(B2)
Unpack	UNPK	F3	SS	D1(L1,B1),D2(L2,B2)
Write Direct (b,p)	WRD	84	SI	D1(B1),I2
Zero and Add (c,d)	ZAP	F8	SS	D1(L1,B1),D2(L2,B2)

NOTES FOR PANELS 1-3

a. Protection feature
b. Direct control feature
c. Condition code is set
d. Decimal feature
e. Model 67
n. New condition
code is loaded
p. Privileged instruction
x. Extended precision floating point feature

MACHINE FORMATS



FLOATING-POINT FEATURE INSTRUCTIONS

Add Normalized, Extended (c,x)	AXR	36	RR	R1,R2
Add Normalized, Long (c)	ADR	2A	RR	R1,R2
Add Normalized, Long (c)	AD	6A	RX	R1,D2(X2,B2)
Add Normalized, Short (c)	AER	3A	RR	R1,R2
Add Normalized, Short (c)	AE	7A	RX	R1,D2(X2,B2)
Add Unnormalized, Long (c)	AWR	2E	RR	R1,R2
Add Unnormalized, Long (c)	AW	6E	RX	R1,D2(X2,B2)
Add Unnormalized, Short (c)	AUR	3E	RR	R1,R2
Add Unnormalized, Short (c)	AU	7E	RX	R1,D2(X2,B2)
Compare, Long (c)	CDR	29	RR	R1,R2
Compare, Long (c)	CD	69	RX	R1,D2(X2,B2)
Compare, Short (c)	CER	39	RR	R1,R2
Compare, Short (c)	CE	79	RX	R1,D2(X2,B2)
Divide, Long	DDR	2D	RR	R1,R2
Divide, Long	DD	6D	RX	R1,D2(X2,B2)
Divide, Short	DER	3D	RR	R1,R2
Divide, Short	DE	7D	RX	R1,D2(X2,B2)
Halve, Long	HDR	24	RR	R1,R2
Halve, Short	HER	34	RR	R1,R2
Load and Test, Long (c)	LTDR	22	RR	R1,R2
Load and Test, Short (c)	LTER	32	RR	R1,R2
Load Complement, Long (c)	LCDR	23	RR	R1,R2
Load Complement, Short (c)	LCER	33	RR	R1,R2
Load, Long	LDR	28	RR	R1,R2
Load, Long	LD	68	RX	R1,D2(X2,B2)
Load Negative, Long (c)	LNDR	21	RR	R1,R2
Load Negative, Short (c)	LNER	31	RR	R1,R2
Load Positive, Long (c)	LPDR	20	RR	R1,R2
Load Positive, Short (c)	LPER	30	RR	R1,R2
Load Rounded, Extended to Long (x)	LRDR	25	RR	R1,R2
Load Rounded, Long to Short (x)	LRER	35	RR	R1,R2
Load, Short	LER	38	RR	R1,R2
Load, Short	LE	78	RX	R1,D2(X2,B2)
Multiply, Extended (x)	MXR	26	RR	R1,R2
Multiply, Long	MDR	2C	RR	R1,R2
Multiply, Long	MD	6C	RX	R1,D2(X2,B2)
Multiply, Long/Extended (x)	MXDR	27	RR	R1,R2
Multiply, Long/Extended (x)	MXD	67	RX	R1,D2(X2,B2)
Multiply, Short	MER	3C	RR	R1,R2
Multiply, Short	ME	7C	RX	R1,D2(X2,B2)
Store, Long	STD	60	RX	R1,D2(X2,B2)
Store, Short	STE	70	RX	R1,D2(X2,B2)
Subtract Normalized, Extended (c,x)	SXR	37	RR	R1,R2
Subtract Normalized, Long (c)	SDR	28	RR	R1,R2
Subtract Normalized, Long (c)	SD	68	RX	R1,D2(X2,B2)
Subtract Normalized, Short (c)	SER	38	RR	R1,R2
Subtract Normalized, Short (c)	SE	78	RX	R1,D2(X2,B2)
Subtract Unnormalized, Long (c)	SWR	2F	RR	R1,R2
Subtract Unnormalized, Long (c)	SW	6F	RX	R1,D2(X2,B2)
Subtract Unnormalized, Short (c)	SUR	3F	RR	R1,R2
Subtract Unnormalized, Short (c)	SU	7F	RX	R1,D2(X2,B2)

NOTES

EXTENDED MNEMONIC INSTRUCTION CODES

GENERAL

Extended Code		Machine Instruction		Meaning
B	D2(X2,B2)	BC 15,	D2(X2,B2)	Branch Unconditionally
BR	R2	BCR 15,	R2	Branch Unconditionally
NOP	D2(X2,B2)	BC 0,	D2(X2,B2)	No Operation
NOPR	R2	BCR 0,	R2	No Operation (RR)

AFTER COMPARE INSTRUCTIONS (A:B)

BH	D2(X2,B2)	BC 2,	D2(X2,B2)	Branch on A High
BL	D2(X2,B2)	BC 4,	D2(X2,B2)	Branch on A Low
BE	D2(X2,B2)	BC 8,	D2(X2,B2)	Branch on A Equal B
BNH	D2(X2,B2)	BC 13,	D2(X2,B2)	Branch on A Not High
BNL	D2(X2,B2)	BC 11,	D2(X2,B2)	Branch on A Not Low
BNE	D2(X2,B2)	BC 7,	D2(X2,B2)	Branch on A Not Equal B

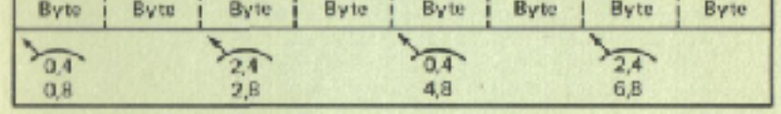
AFTER ARITHMETIC INSTRUCTIONS

BO	D2(X2,B2)	BC 1,	D2(X2,B2)	Branch on Overflow
BP	D2(X2,B2)	BC 2,	D2(X2,B2)	Branch on Plus
BM	D2(X2,B2)	BC 4,	D2(X2,B2)	Branch on Minus
BZ	D2(X2,B2)	BC 8,	D2(X2,B2)	Branch on Zero
BNP	D2(X2,B2)	BC 13,	D2(X2,B2)	Branch on Not Plus
BNM	D2(X2,B2)	BC 11,	D2(X2,B2)	Branch on Not Minus
BNZ	D2(X2,B2)	BC 7,	D2(X2,B2)	Branch on Not Zero

AFTER TEST UNDER MASK INSTRUCTIONS

BO	D2(X2,B2)	BC 1,	D2(X2,B2)	Branch if Ones
BM	D2(X2,B2)	BC 4,	D2(X2,B2)	Branch if Mixed
BZ	D2(X2,B2)	BC 8,	D2(X2,B2)	Branch if Zeros
BNO	D2(X2,B2)	BC 14,	D2(X2,B2)	Branch if Not Ones

CNOP ALIGNMENT



EDIT AND EDMK PATTERN CHARACTERS (in hex)

20–digit selector	40–blank	5C–asterisk
21–start of significance	4B–period	6B–comma
22–field separator	5B–dollar sign	C3D9–CR

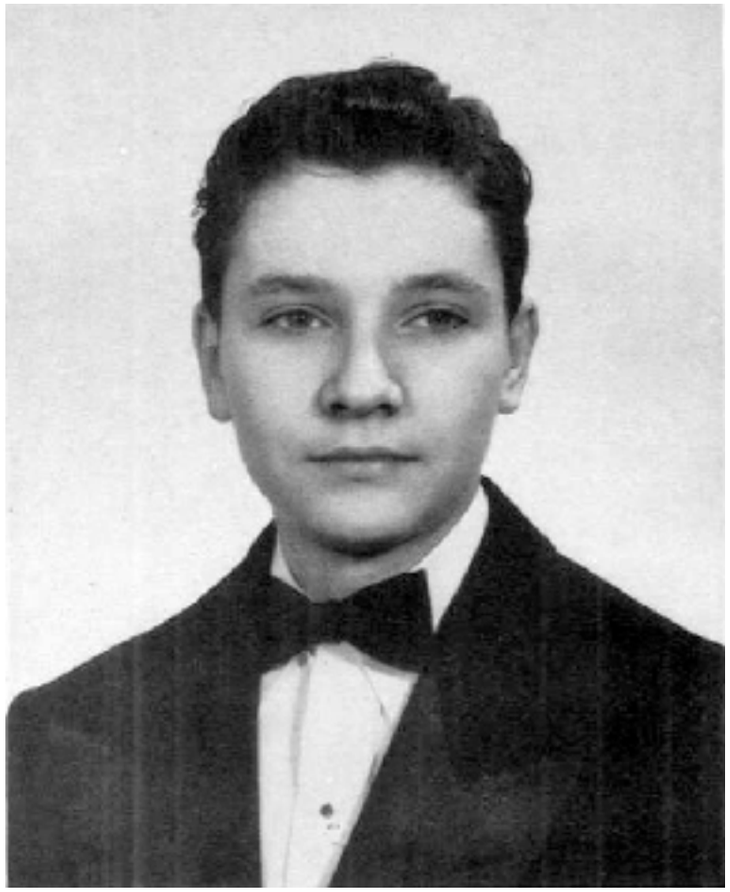
SUMMARY OF CONSTANTS (OS and DOS Assemblers)

TYPE	IMPLIED LENGTH, BYTES	ALIGNMENT	FORMAT	TRUNCATION/ PADDING
C	–	byte	characters	right
X	–	byte	hexadecimal digits	left
B	–	byte	binary digits	left
F	4	word	fixed-point binary	left
H	2	halfword	fixed-point binary	left
E	4	word	short floating-point	right
D	8	doubleword	long floating-point	right
L	16	doubleword	extended floating-point	right
P	–	byte	packed decimal	left
Z	–	byte	zoned decimal	left
A	4	word	value of address	left
Y	2	halfword	value of address	left
S	2	halfword	address in base-displacement form	–
V	4	word	externally defined address value	left
Q*	4	word	symbol naming a DXD or DSECT	left

*OS only

Innovations?

- Out of order execution (Tomasulo's Algorithm)
- Page-based virtual addressing (memory)
- Availability of a family of I/O devices
- Architecture as we know it
    - separation of microarchitecture and architecture
    - or implementation and ISA
- Operating system
- Shared storage device
- The byte.



Robert Tomasulo

Tomasulo's Algorithm

- Uncovered instruction level parallelism
- for (i = 0; ; i++) a[i] = a[i] + r2;
- HERE:
  LOAD @(r4) - > R1
  ADD R1, R2 -> R1
  STORE R1 -> @(r4)
  ADD #1, R4 -> R4
  JUMP HERE

- LOAD @(r4) - > R1
  ADD R1, R2 -> R1
  STORE R1 -> @(r4)
  ADD #1, R4 -> R4
  LOAD @(r4) - > R1
  ADD R1, R2 -> R1
  STORE R1 -> @(r4)
  ADD #1, R4 -> R4
  JUMP HERE

MAP: R1:P1, R2:P2, R3:P3, R4:P4
FREE-LIST: P5,P6,P7,P8, ….

- LOAD @(r4) - > R1
  ADD R1, R2 -> R1
  STORE R1 -> @(r4)
  ADD #1, R4 -> R4
  LOAD @(r4) - > R1
  ADD R1, R2 -> R1
  STORE R1 -> @(r4)
  ADD #1, R4 -> R4
  JUMP HERE

LOAD @(P4),P5
ADD P5, P2, P6
STORE P6, @(P4)
ADD #1, P4, P7
LOAD @(P7),P8

MAP: R1:P5, R2:P2, R3:P3, R4:P4
FREE-LIST: P6,P7,P8, ….
MAP: R1:P6, R2:P2, R3:P3, R4:P4
FREE-LIST:P7,P8, ….
MAP: R1:P6, R2:P2, R3:P3, R4:P4
FREE-LIST:P7,P8, ….
MAP: R1:P6, R2:P2, R3:P3, R4:P7
FREE-LIST P8, ….
MAP: R1:P8, R2:P2, R3:P3, R4:P7
FREE-LIST, ….

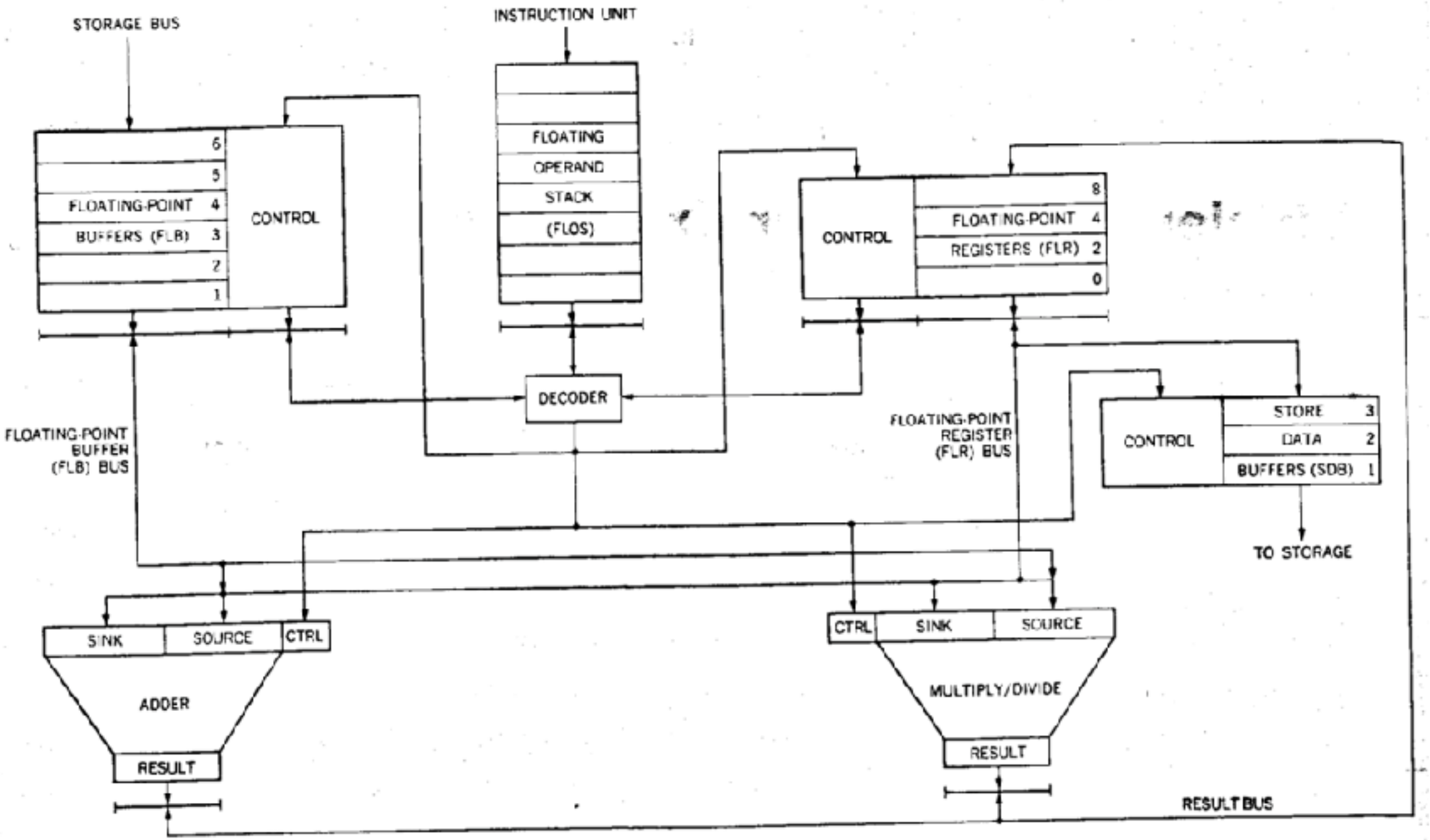


Figure 1 Data registers and transfer paths without CDB.

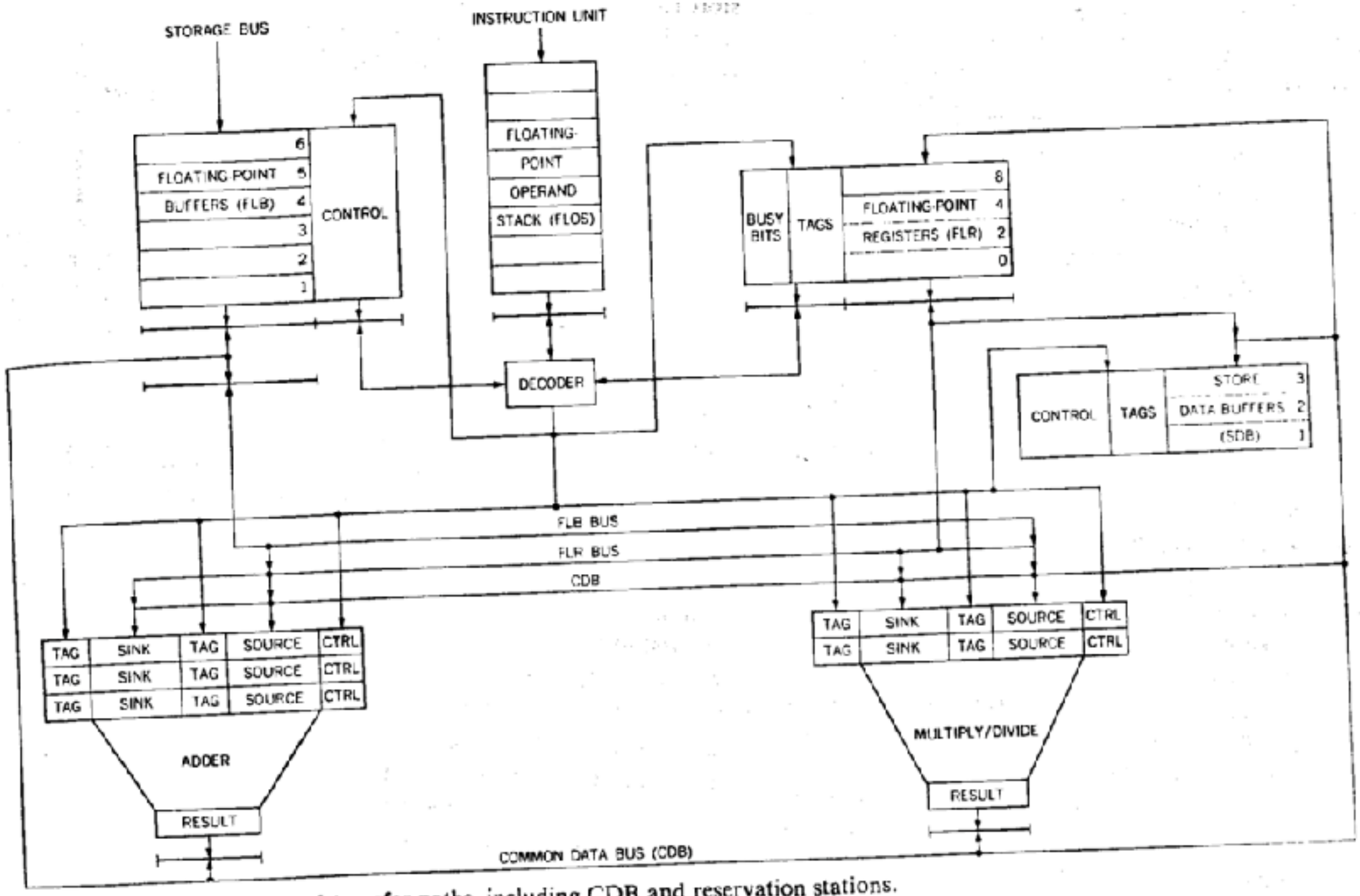
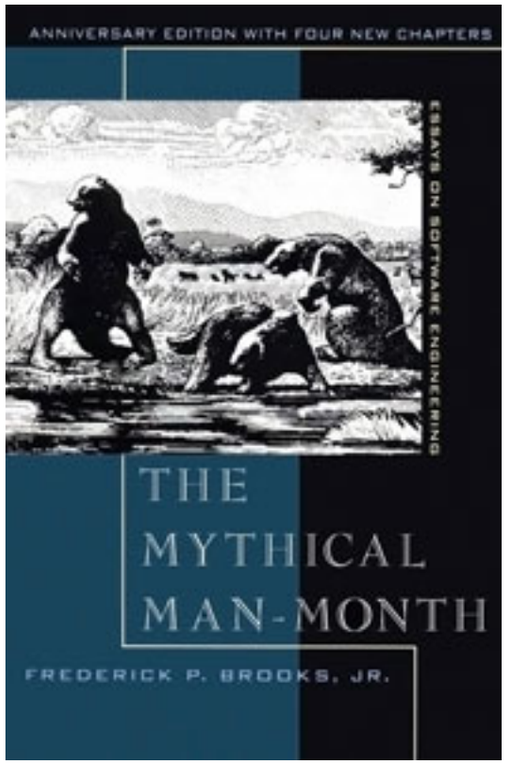
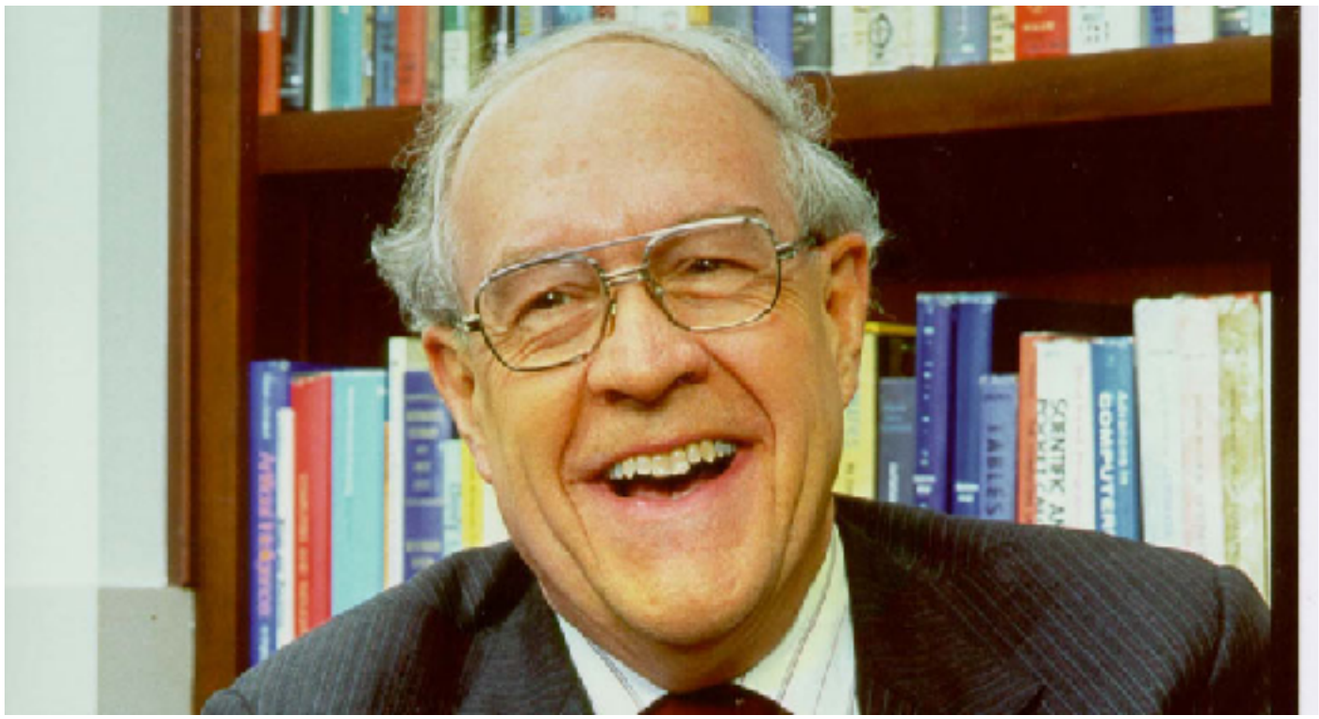


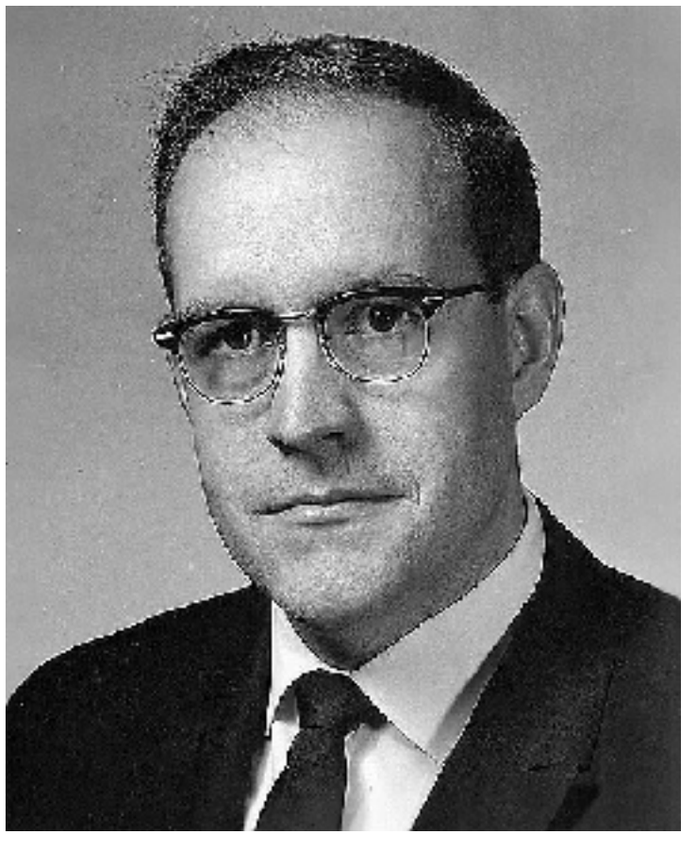
Figure 4 Data registers and transfer paths, including CDB and reservation stations.

Troubles with Out of Order Execution ala Tomasulo's Alg.

- Big
- Out of execution leads to exception problems
- Matching software to hardware resources


ANNIVERSARY EDITION WITH FOUR NEW CHAPTERS
ESSAYS ON SOFTWARE ENGINEERING
THE
MYTHICAL
MAN-MONTH
FREDERICK P. BROOKS, JR.





Fred Brooks