ELEC 5200-001/6200-001
Computer Architecture and Design
Fall 2013
History of Computers (Chapter 1)

Vishwani D. Agrawal
James J. Danaher Professor
Victor P. Nelson
Professor
Department of Electrical and Computer Engineering
Auburn University, Auburn, AL 36849
http://www.eng.auburn.edu/~vagrawal
vagrawal@eng.auburn.edu

Historic Events

- 1623, 1642: Wilhelm Schickard (1592-1635) and Blaise Pascal (1623-1662) built mechanical counters with carry.
- 1823-34: Charles Babbage designed a difference engine. http://www.youtube.com/watch?v=0anIyVGeWOI&feature=related
- 1941: Conrad Zuse (1910-1995) built Z3, the first working programmable computer, built in Germany.

Z3

Conrad Zuse

Z1 (1938)

Z2 (1939)

Z3 (1941)

.

.

.

Historic Events

- 1942: Vincent Atanasoff (professor) and Clifford Barry (graduate assistant) built the first electronic computer (ABC) at Iowa State College.
- 1943-44: John Mauchly (professor) and J. Presper Eckert (graduate student) built ENIAC at U. Pennsylvania. 1623,
- 1944: Howard Aiken used "separate data and program memories" in MARK I – IV computers – *Harvard Architecture*.
- 1945-52: John von Neumann proposed a "*stored program computer*" EDVAC (Electronic Discrete Variable Automatic Computer) – *Von Neumann Architecture* – use the same memory for program and data.

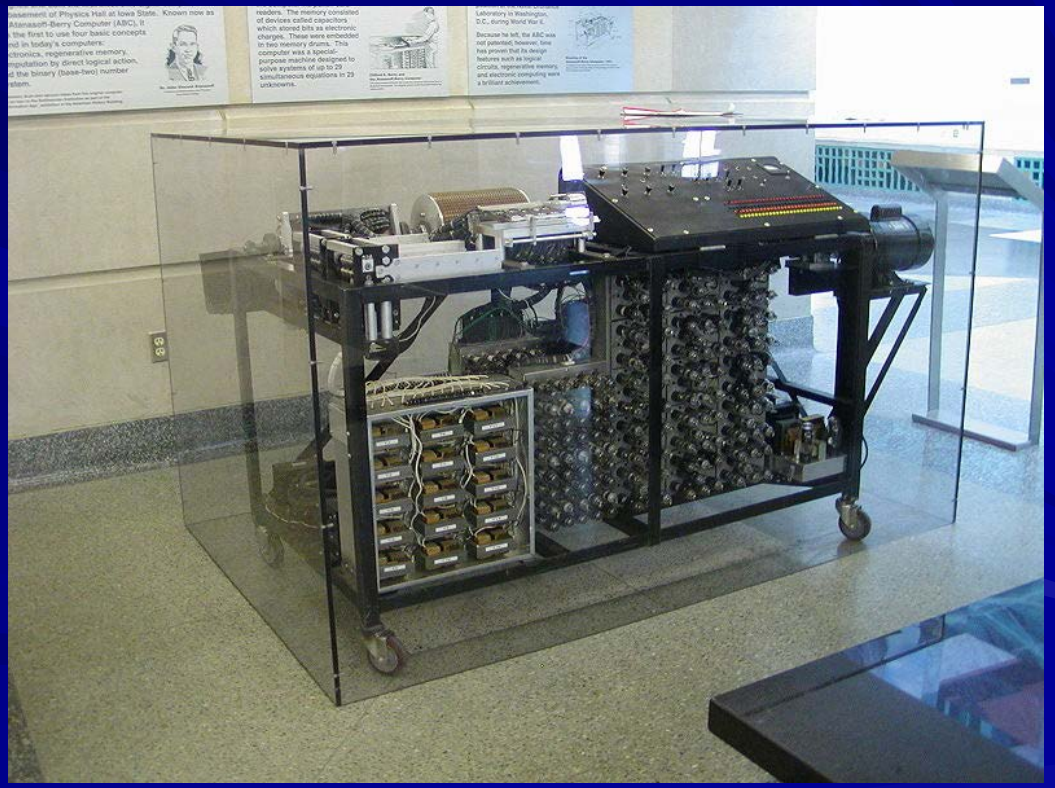
The Atanasoff Story

- *The First Electronic Computer, the Atanasoff Story*, by Alice R. Burks and Arthur W. Burks, Ann Arbor, Michigan: The University of Michigan Press, 1991.

- *The Man Who Invented the Computer: The Biography of John Atanasoff, Digital Pioneer*, by Jane Smiley, 256 pages, Doubleday, $25.95.

National Medal of Technology 1990

John Vincent Atanasoff (1903–1995)



Most Influential Document

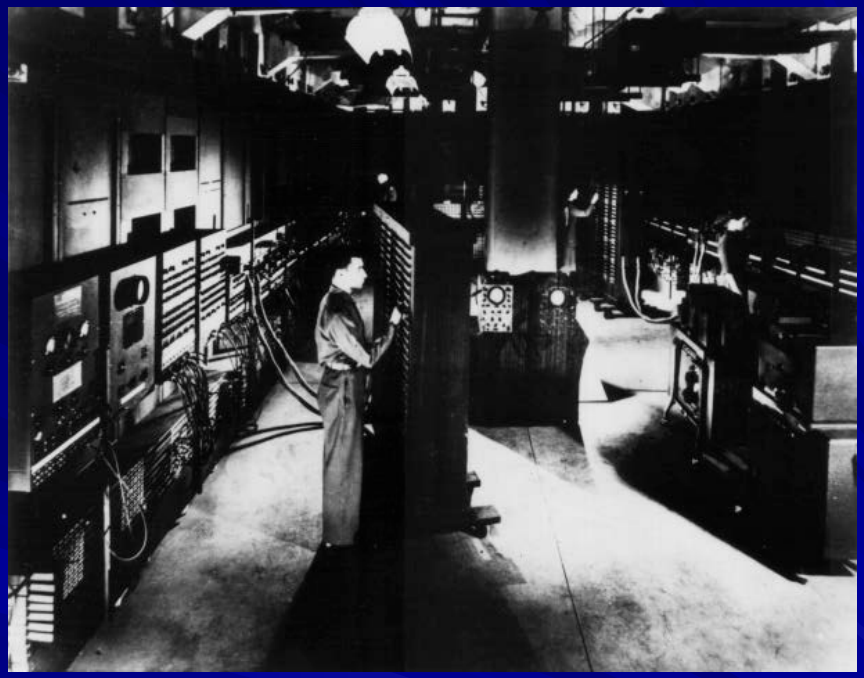
- "Preliminary Discussion of the Logical Design of an Electronic Computing Instrument," 1946 report by A. W. Burks, H. H. Holdstine and J. von Neumann. Appears in *Papers of John von Neumann*, W. Aspray and A. Burks (editors), MIT Press, Cambridge, Mass., 1987, pp. 97-146.

History Continues

- 1946-52: Von Neumann built the IAS computer at the Institute of Advanced Studies, Princeton – ***A prototype for most future computers*.**
- 1947-50: Eckert-Mauchly Computer Corp. built UNIVAC I (Universal Automatic Computer), used in the 1950 census.
- 1949: Maurice Wilkes built EDSAC (Electronic Delay Storage Automatic Calculator), the first stored-program computer.

First General-Purpose Computer

- Electronic Numerical Integrator and Calculator (ENIAC) built in World War II was the first general purpose computer
  - Used for computing artillery firing tables
  - 80 feet long, 8.5 feet high and several feet wide
  - Twenty 10 digit registers, each 2 feet long
  - Multiply/divide/sq-root
  - Used 18,000 vacuum tubes
  - 5,000 additions/second
  - Weight: 30 tons
  - Power consumption: 140kW
  - Data input: 800 card/min reader



EDSAC
Electronic Discrete Variable Computer

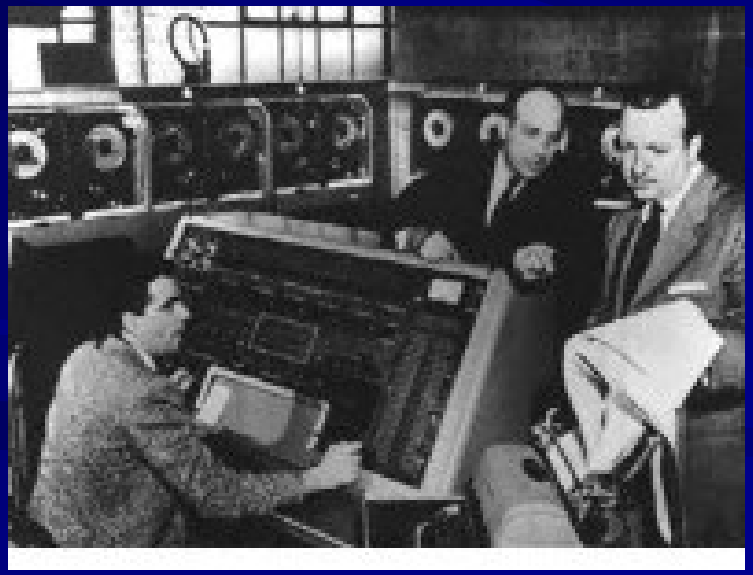
- 1945-52, Jon Von-Neumann *(not built)*
- Programs+data in one memory
- Mercury delay-line memory
- 44-bit binary numbers, serial arithmetic
- Arithmetic instruction: A1 A2 A3 A4 op next

  A1 op A2 -> A3, then goto next
- Conditional jump: A1 A2 A3 A4 C

  if *A1 ≥ A2 goto A3, else goto A4*

Univac: Election, Nov. 4, 1952

Candidate	Electoral votes	
	Univac prediction	Actual count
Eisenhower	438	442
Stevenson	93	89



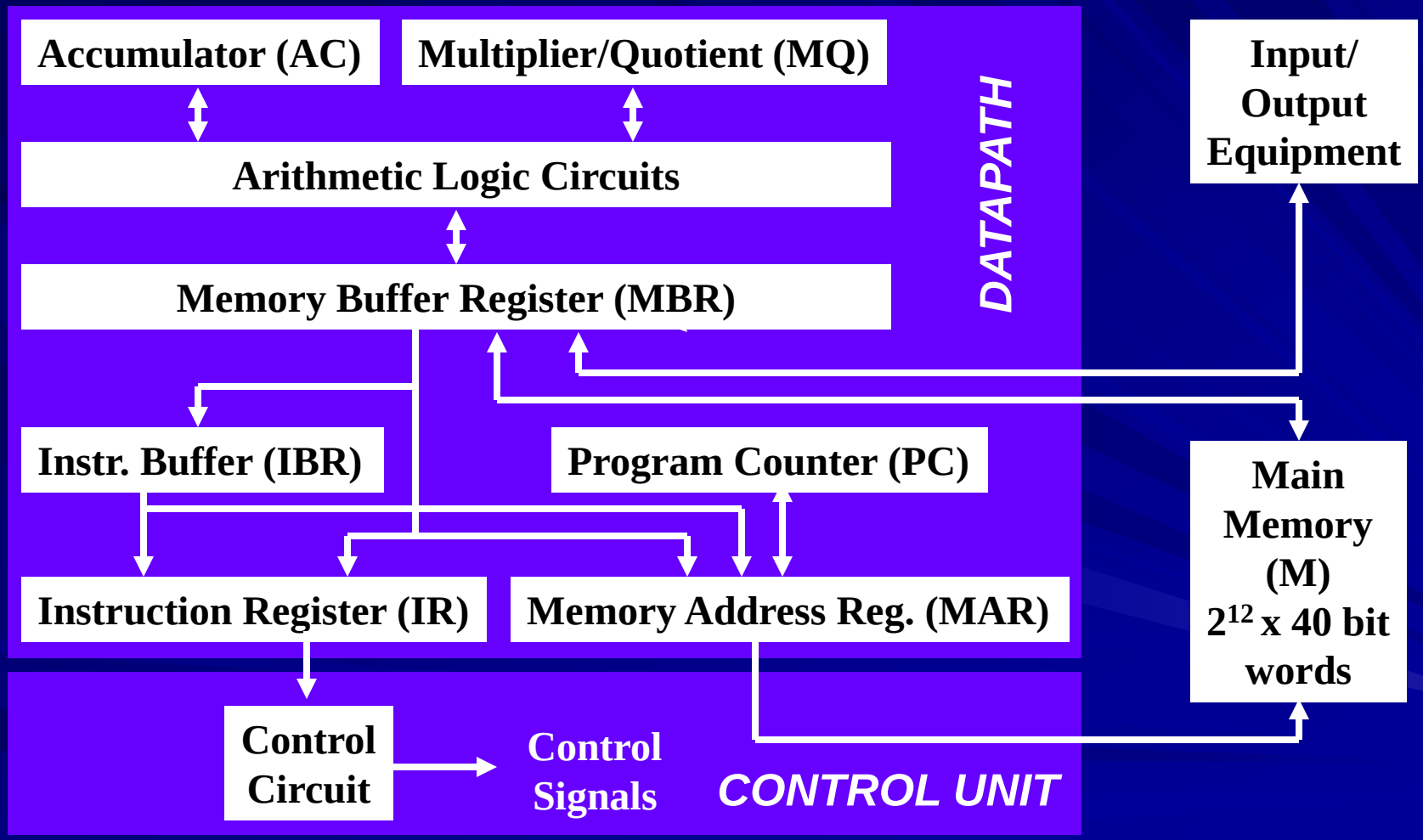
Harold Sweeney, operator
J. Presper Eckert, co-inventor
Walter Cronkite, CBS

USA TODAY, Oct 27, 2004, p. B.3

First-Generation Computers

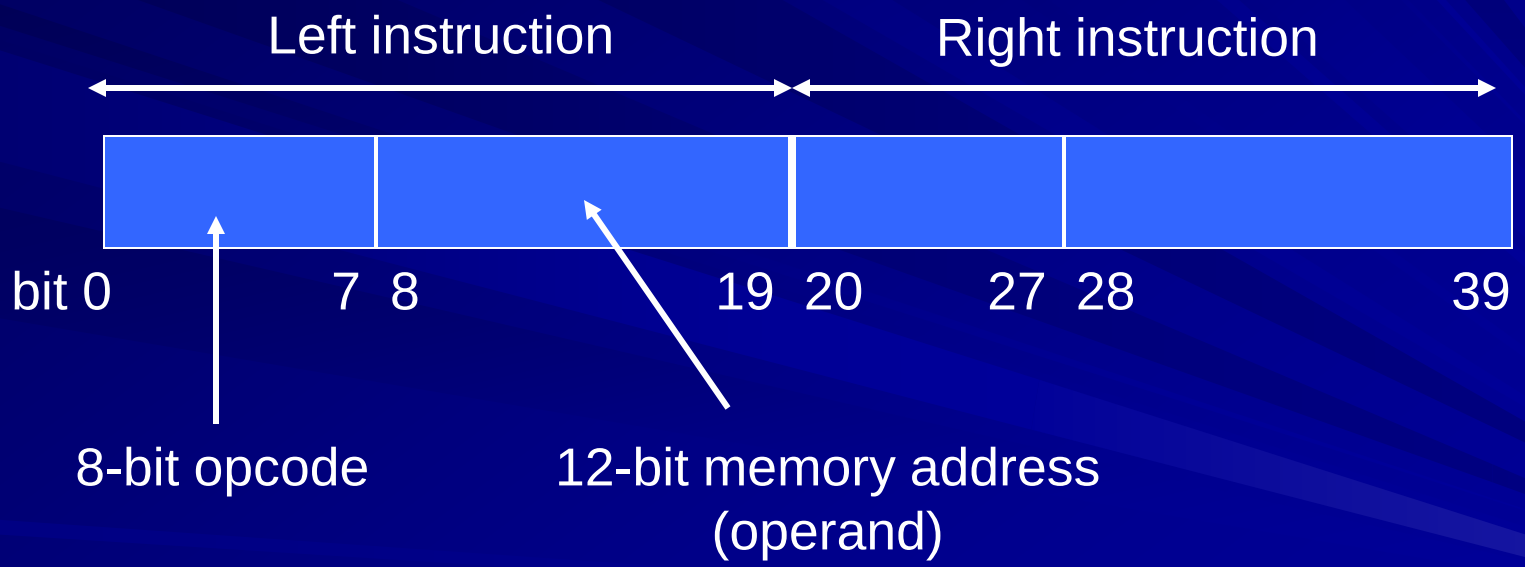
- Late 1940s and 1950s
- Stored-program computers
- Programmed in assembly language
- Used magnetic devices and earlier forms of memories
- Examples: IAS, ENIAC, EDVAC, UNIVAC, Mark I, IBM 701

The Organization of IAS Computer



IAS Computer Machine Language

40-bit word, two machine instructions per word.



Ref: **J. P. Hayes, *Computer Architecture and Organization*, New York: McGraw-Hill, 1978.**

IAS Data Transfer Instructions (7)

Instruction	*Opcode*	*Description*
LOAD MQ	00001010	AC ← MQ
LOAD MQ, M(X)	00001001	MQ ← M(X)
STOR M(X)	00100001	M(X) ← AC
LOAD M(X)	00000001	AC ← M(X)
LOAD – M(X)	00000010	AC ← – M(X)
LOAD |M(X)|	00000011	AC ← |M(X)|
LOAD – |M(X)|	00000100	AC ← – |M(X)|

Ref. **W. Stallings, *Computer Organization & Architecture, Sixth Edition*, Prentice-Hall, 2003, page 23.**

IAS Unconditional Branch Instructions (2)

Instruction	*Opcode*	*Description*
JUMP M(X,0:19)	00001101	next instruction M(X,0:19)
JUMP M(X,20:39)	00001110	next instruction M(X,20:39)

IAS Conditional Branch Instructions (2)

Instruction	*Opcode*	*Description*
JUMP +M(X,0:19)	00001111	IF AC ≥ 0, then next instruction M(X,0:19)
JUMP +M(X,20:39)	00010000	IF AC ≥ 0, then next instruction M(X,20:39)

IAS Arithmetic Instructions (8)

Instruction	*Opcode*	*Description*
ADD M(X)	00000101	AC ← AC + M(X)
ADD |M(X)|	00000111	AC ← AC + |M(X)|
SUB M(X)	00000110	AC ← AC – M(X)
SUB |M(X)|	00001000	AC ← AC – |M(X)|
MUL M(X)	00001011	AC, MQ ← MQ×M(X)
DIV M(X)	00001100	MQ, AC ← MQ/M(X)
LSH	00010100	AC ← AC x 2
RSH	00010101	AC ← AC / 2

IAS Address Modify Instructions (2)

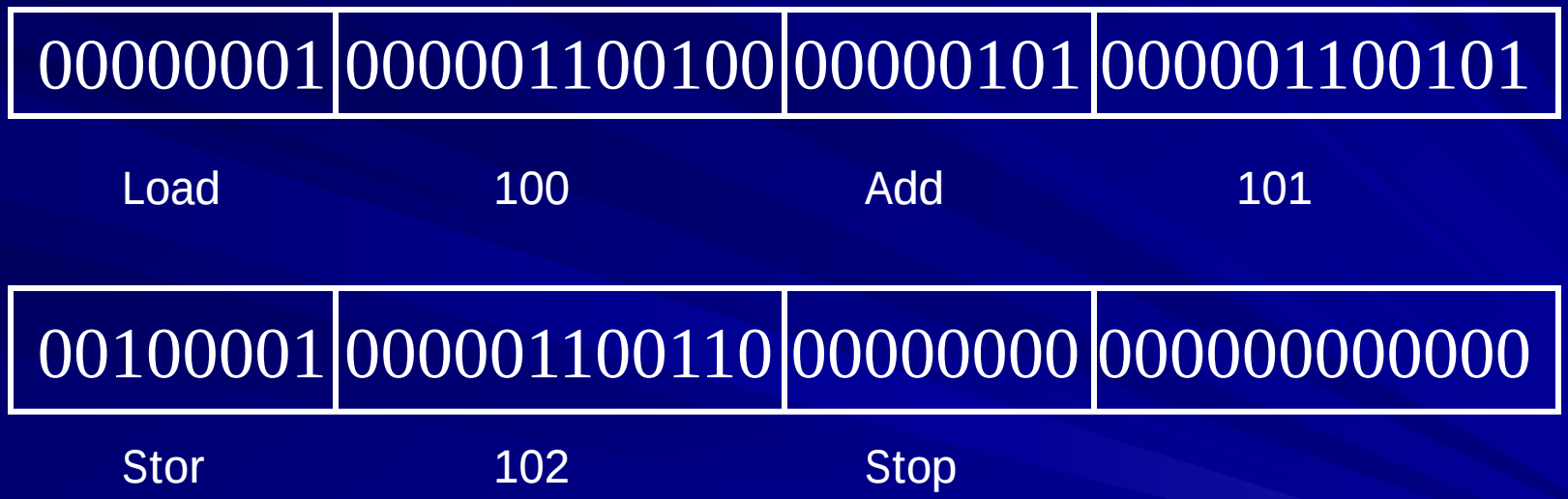
Instruction	*Opcode*	*Description*
STOR M(X,8:19)	00010010	M(X,8:19) ← AC(28:39)
STOR M(X,28:39)	00010011	M(X,28:39) ← AC(28:39)

How IAS Computer Adds Two Numbers

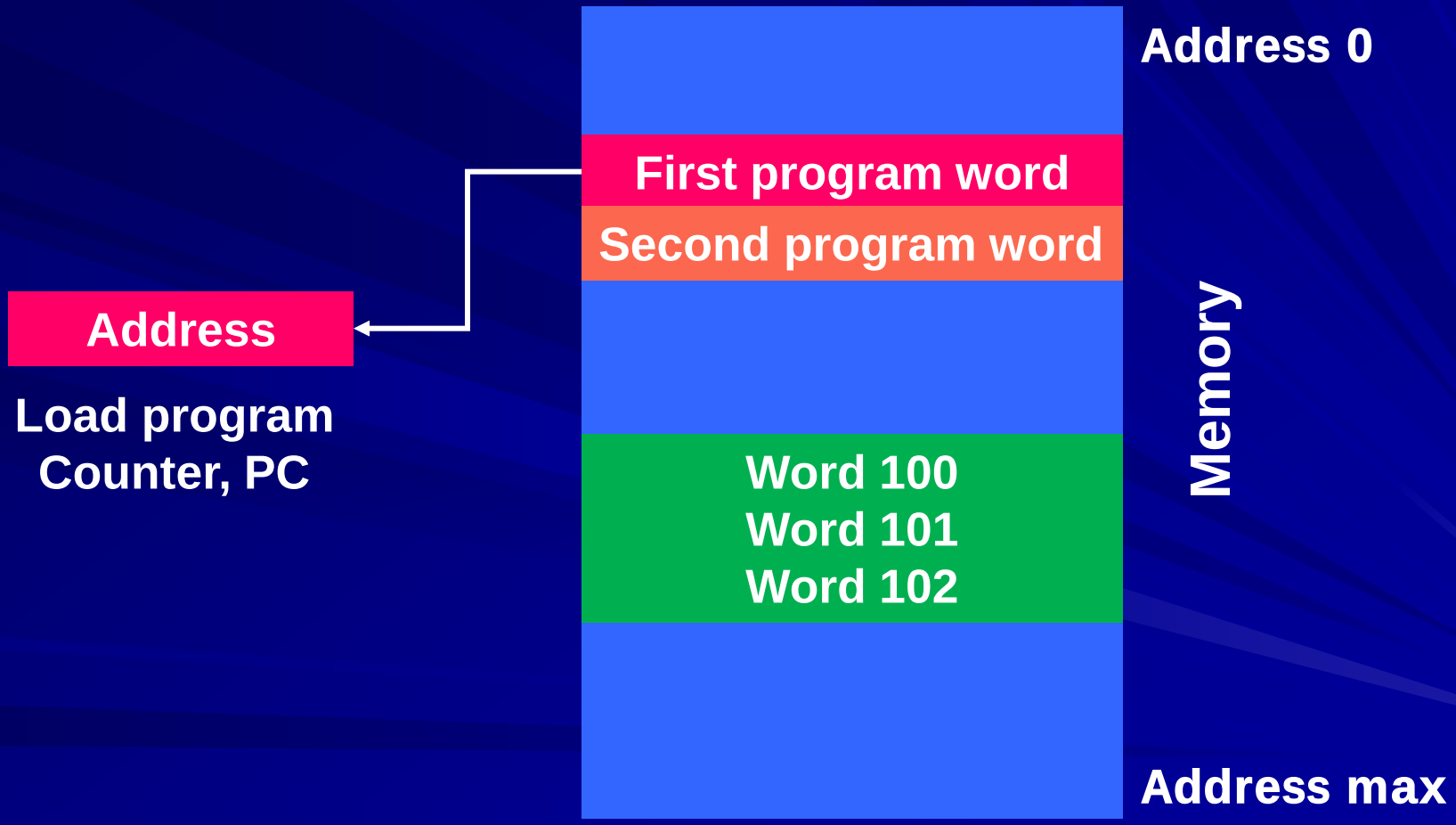
- Suppose the numbers are stored in memory locations 100 and 101, and
- The sum is to be saved in memory location 102

Instruction	***Opcode***	***Description***
LOAD M(100)	00000001	AC ← M(100)
ADD M(101)	00000101	AC ← AC + M(101)
STOR M(102)	00100001	M(102) ← AC

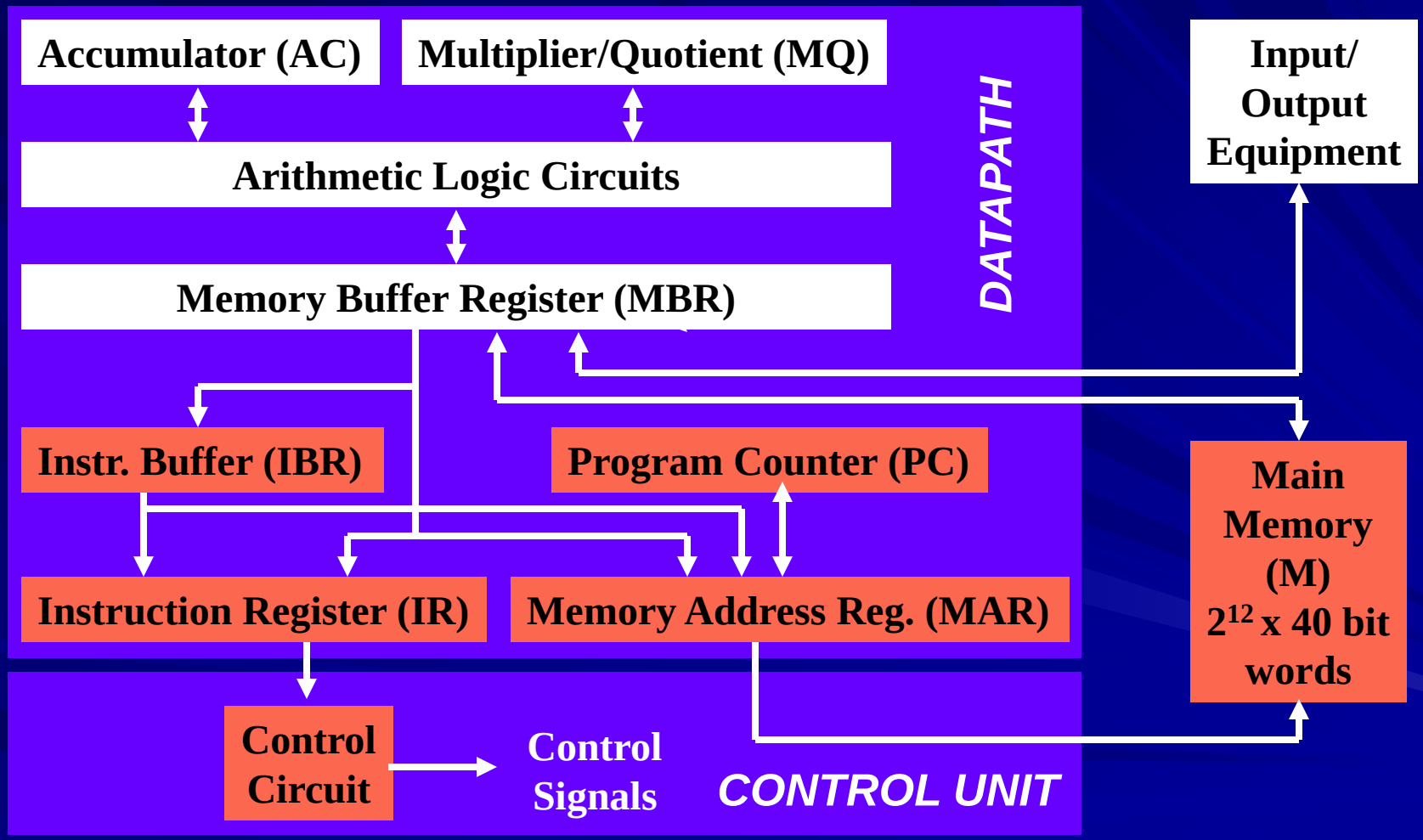
IAS Computer Machine Code



Save Program in Memory



Executing the Program



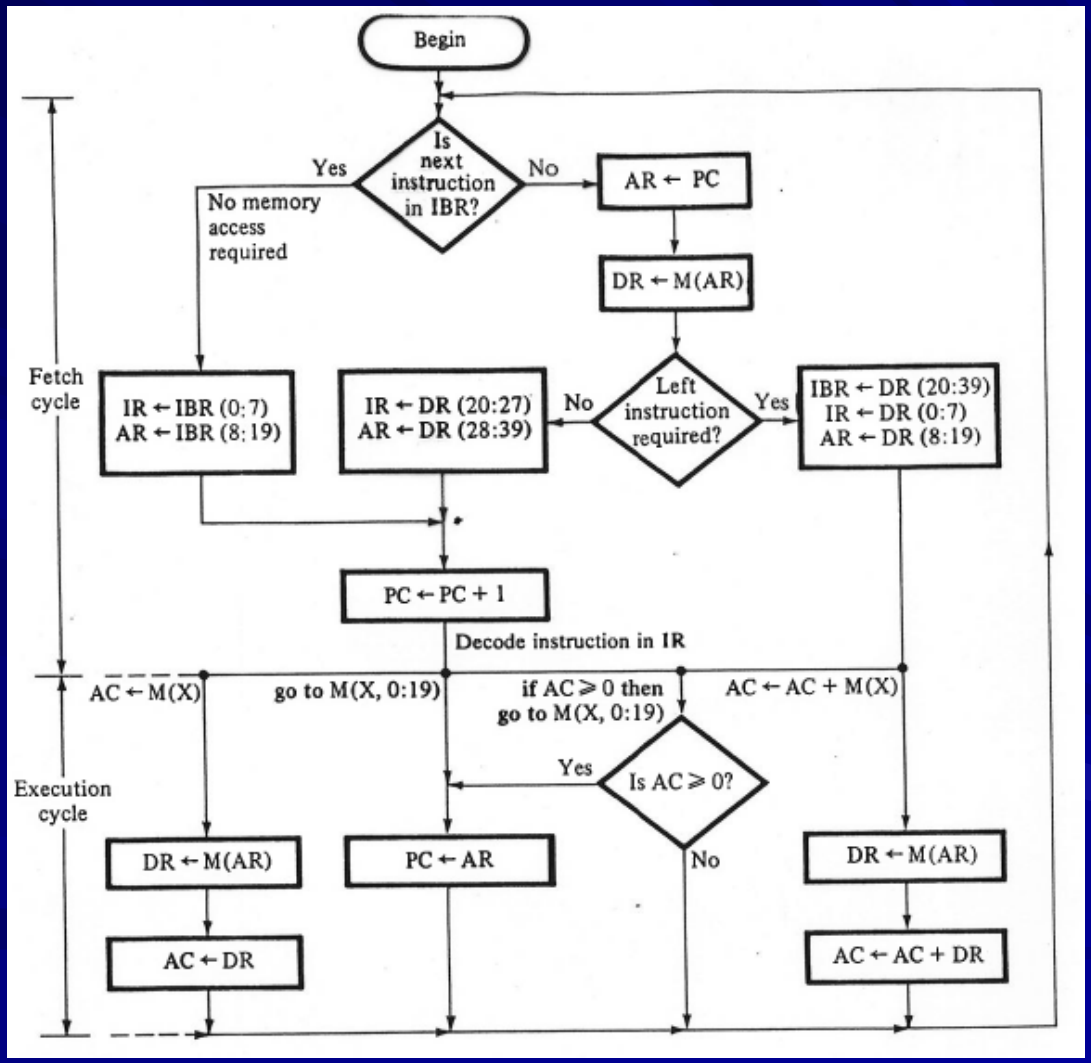
IAS Instruction Cycles

- Store machine code in contiguous words of memory.
- Place starting address in program counter (PC).
- Start program: MAR ← PC
- Read memory: IBR ← MBR ← M(MAR), *fetch*
- Place left instruction (Load) in IR and operand (address) 100 in MAR, *decode*
- Read memory: AC ← M(100), *execute*
- Place right instruction (Add) in IR and operand (address) 101 in MAR, *decode*
- Read memory and add: AC ← AC + M(101), *execute*
- PC ← PC + 1

IAS Instruction Cycles (Cont.)

- MAR ← PC
- Read memory: IBR ← MBR ← M(MAR), *fetch*
- Place left instruction (Stor) in IR and operand (address) 102 in MAR, *decode*
- MBR ← AC, *execute*
- Write memory
- Place right instruction (Stop) in IR and operand 000 in MAR, *decode*
- Stop, *execute*

IAS Control Flowshart



Sample IAS Program (1)

Instruction	Comments
MQ ← M(1)	Transfer a_{12} to MQ
AC.MQ ← MQ × M(3)	Compute $a_{12}a_{21}$
M(11) ← AC	Transfer $a_{12}a_{21}$ to M(11)
MQ ← M(0)	Transfer a_{11} to MQ
AC.MQ ← MQ × M(4)	Compute $a_{11}a_{22}$
AC ← AC − M(11)	Compute $a_{11}a_{22} - a_{12}a_{21}$
M(13) ← AC	Transfer $a_{11}a_{22} - a_{12}a_{21}$ to M(13)
MQ ← M(1)	Transfer a_{12} to MQ
AC.MQ ← MQ × M(5)	Compute $a_{12}b_2$
M(9) ← AC	Transfer $a_{12}b_2$ to M(9)
MQ ← M(4)	Transfer a_{22} to MQ
AC.MQ ← MQ × M(2)	Compute $a_{22}b_1$
AC ← AC − M(9)	Compute $a_{22}b_1 - a_{12}b_2$
MQ.AC ← AC ÷ M(13)	Compute x_1
M(14) ← AC	Transfer remainder of x_1 to M(14)
AC ← MQ	Transfer quotient of x_1 to AC
M(15) ← AC	Transfer quotient of x_1 to M(15)

$$x_1 = \frac{(a_{22}b_1 - a_{12}b_2)}{a_{11}a_{22} - a_{12}a_{21}}$$

FIGURE 1.18
An IAS version of Menabrea's program.

Sample IAS program (2)

Location	Instruction or data	Comments
0	999	Count N
1	1	Constant
2	1000	Constant
3L	AC ← M(2000)	Transfer A(I) to AC
3R	AC ← AC + M(3000)	Compute A(I) + B(I)
4L	M(4000) ← AC	Transfer sum to C(I)
4R	AC ← M(0)	Load count N
5L	AC ← AC − M(1)	Decrement N by 1
5R	**if** AC ≥ 0 **then go to** M(6, 20:39)	Test N
6L	**go to** M(6, 0:19)	Halt
6R	M(0) ← AC	Update N
7L	AC ← AC + M(1)	Increment AC by 1
7R	AC ← AC + M(2)	
8L	M(3, 8:19) ← AC(28:39)	Modify address in 3L
8R	AC ← AC + M(2)	
9L	M(3, 28:39) ← AC(28:39)	Modify address in 3R
9R	AC ← AC + M(2)	
10L	M(4, 8:19) ← AC(28:39)	Modify address in 4L
10R	**go to** M(3, 0:19)	

C(I) = A(I) + B(I)

FIGURE 1.19
An IAS program for vector addition.

Hardware Contains

- Data storage devices
  - Memory
  - Registers
- Instruction decoding and execution devices
  - Execution unit (arithmetic logic unit, ALU)
  - Data transfer buses
  - Control unit

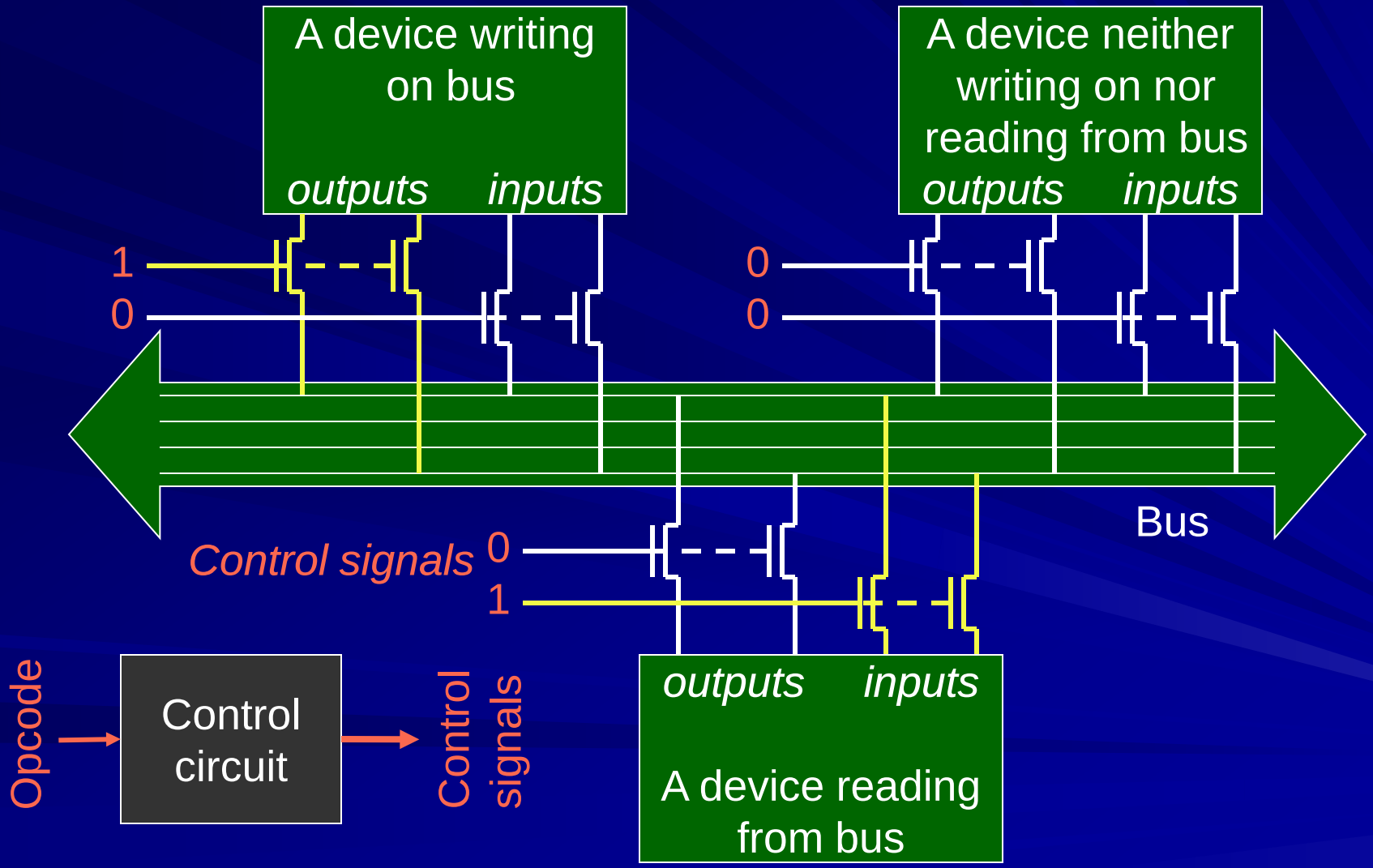
Registers in IAS

Register	Size (bits)	Function
Program counter (PC)	12	Holds mem. address of next instruction
Accumulator (AC)	40	Temporary data storage
Multiplier quotient (MQ)	40	Temporary data storage
Memory buffer (MBR)	40	Memory read / write data
Instruction buffer (IBR)	20	Holds right instr. (bits 20-39)
Instruction register (IR)	8	Holds opcode part of instruction
Memory address (MAR)	12	Holds mem. address part of instruction

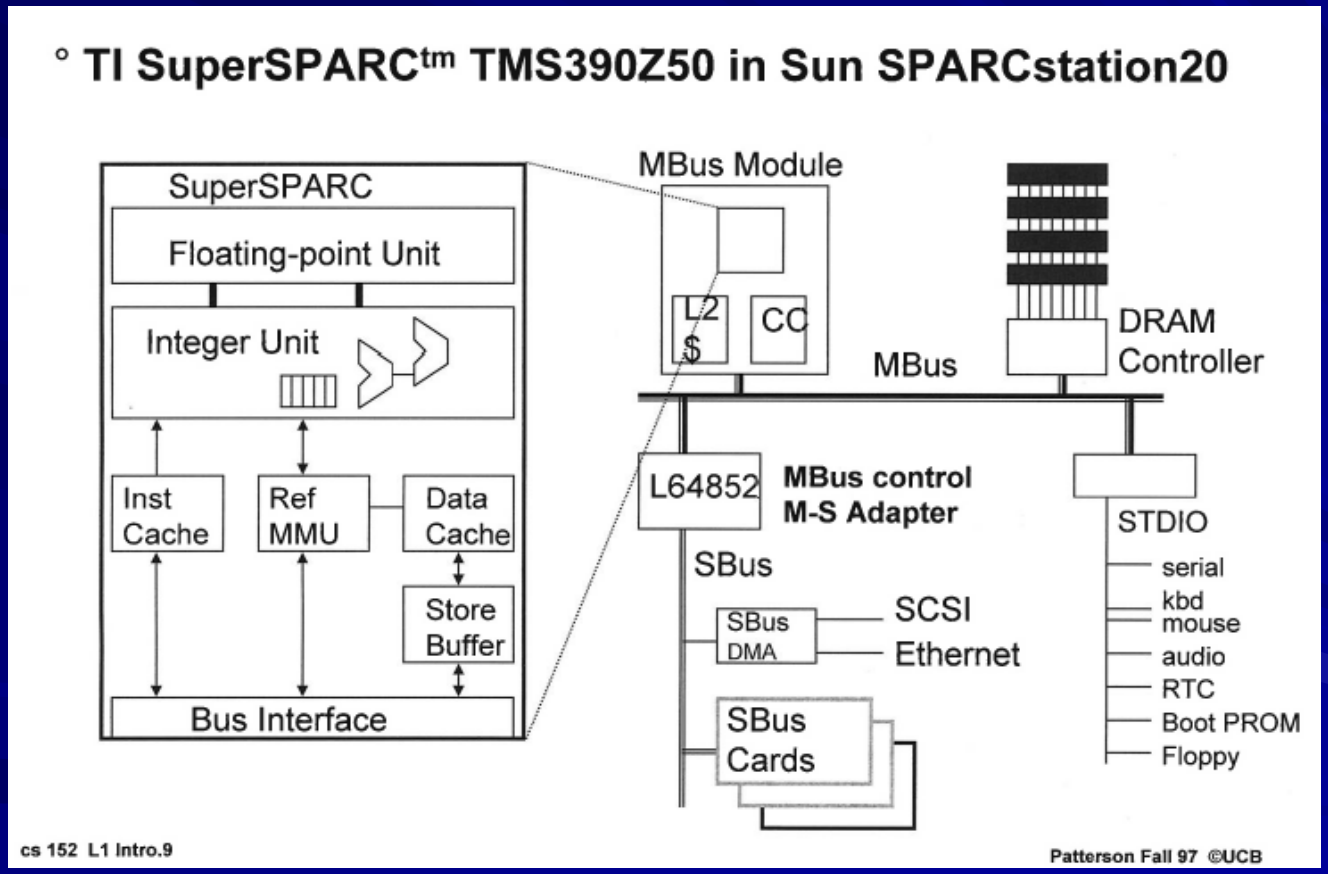
Register Transfer

- Transfer data synchronously with clock
  - Register to register
  - Register to register through ALU logic
  - Registers to register through memory (write)
  - Register to register through memory (read)
- Data transfer through communication bus
  - Source register writes on bus
  - Destination register reads from bus
  - Control circuit provides read / write signals for bus and memory

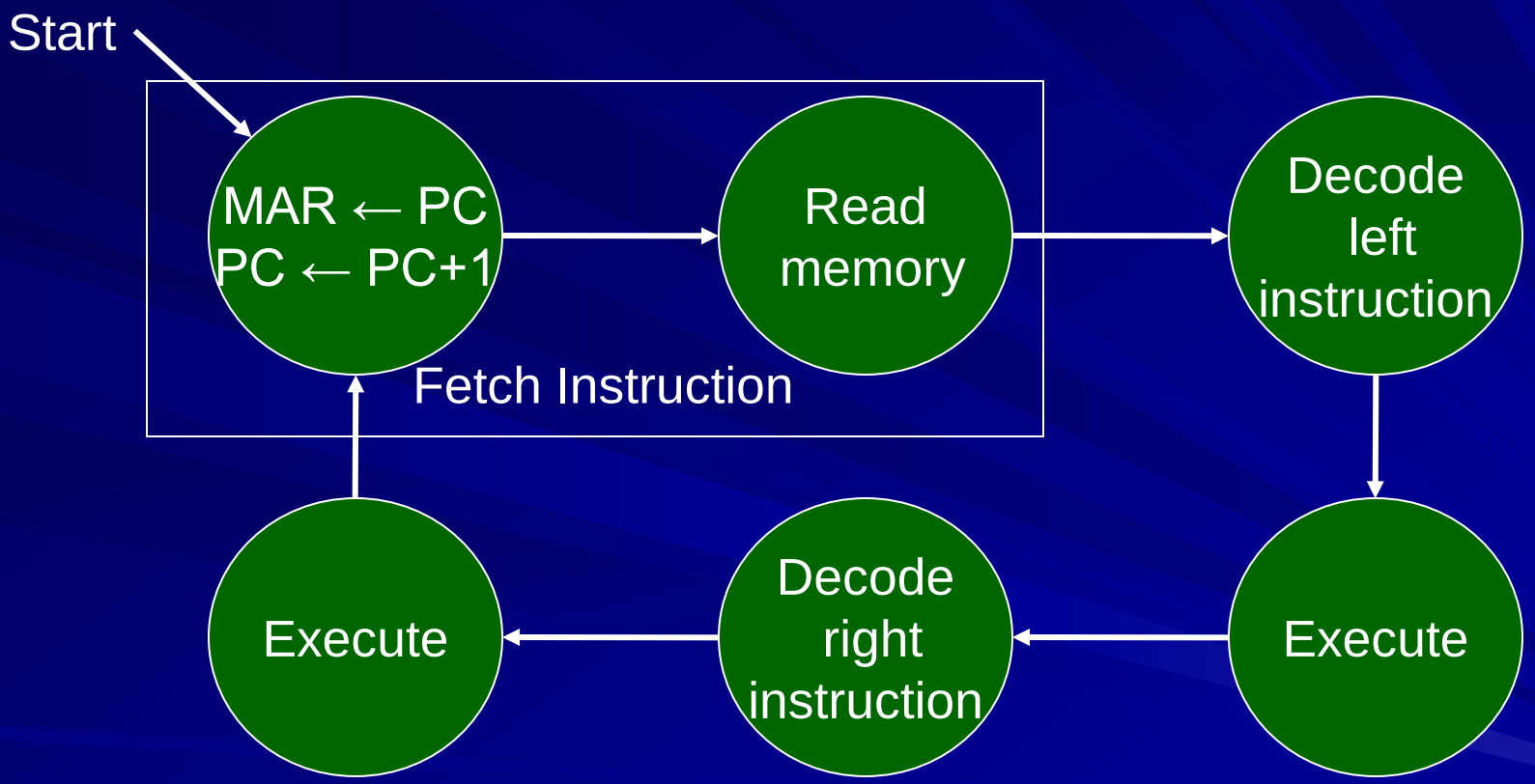
Communication Bus



Multi-bus system



Control Circuit – Finite State Machine

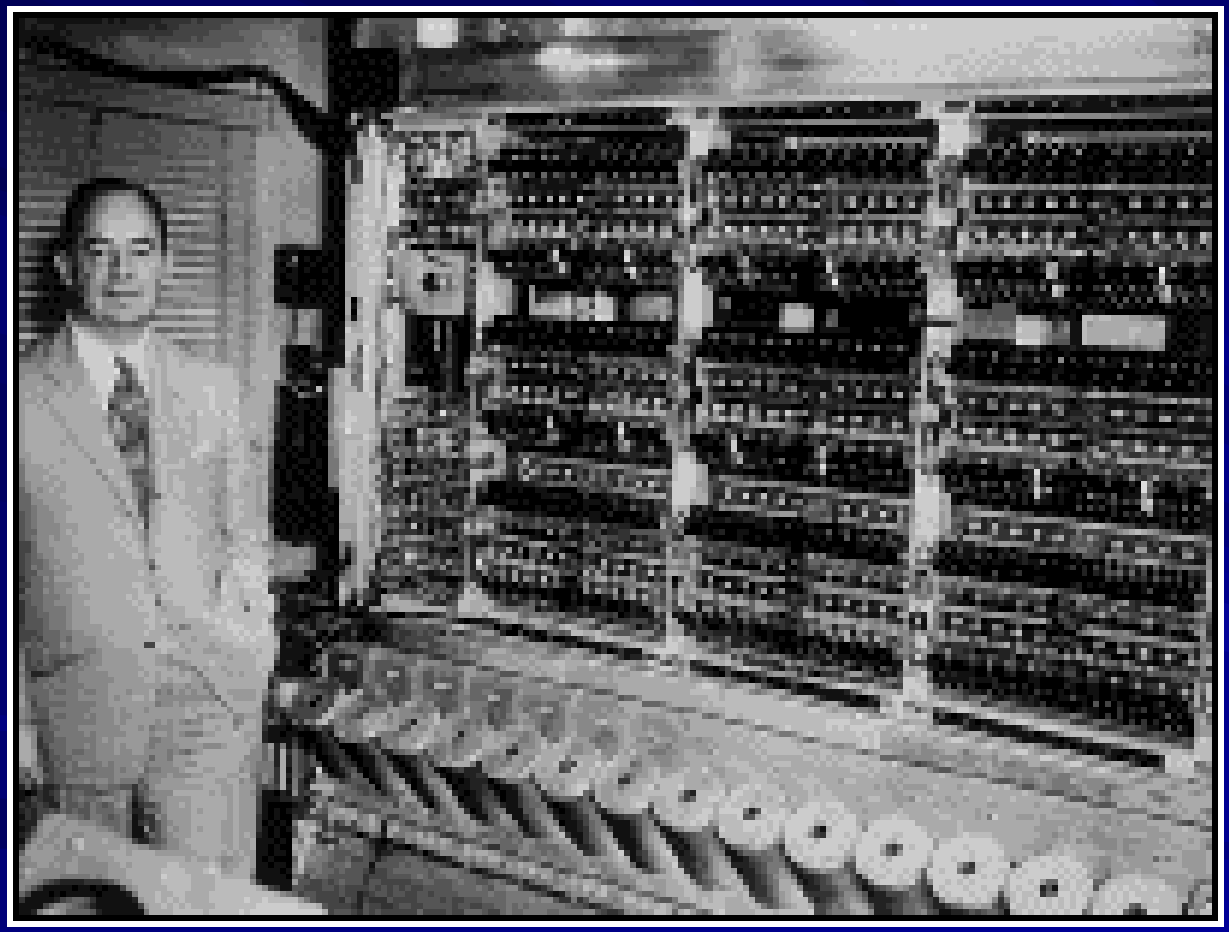


Von Neumann Bottleneck

- Von Neumann architecture uses the same memory for instructions (program) and data.
- The time spent in memory accesses can limit the performance. This phenomenon is referred to as *von Neumann bottleneck*.
- To avoid the bottleneck, later architectures restrict most operands to registers (temporary storage in processor).

Ref.: D. E. Comer, *Essentials of Computer Architecture*, Upper Saddle River, NJ: Pearson Prentice-Hall, 2005, p. 87.

John von Neumann (1903-1957)



Second Generation Computers

- 1955 to 1964
- Transistor replaced vacuum tubes
- Magnetic core memories
- Floating-point arithmetic
- High-level languages used: ALGOL, COBOL and FORTRAN
- System software: compilers, subroutine libraries, batch processing
- Example: IBM 7094

IBM 7094 Computer

- ADDITIONS (to IAS instr. set)
  - Index-register addressing.
  - Logical op's.
  - Floating-Point Op's.
  - I/O Op's
  - More Conditional Jumps

- VECTOR ADDITION PROGRAM

Loc	Op	Addr, Tag, Decr	
	AXT	0,2	(0 → XR(2))
START:	CLA	2000,2	Clear ACC, Add M(2000-XR(2))
	ADD	3000,2	AC + M(3000-XR(2))
	STO	4000,2	AC → M(4000-XR(2))
	TXI	TEST,2,1	XR(2)+1 → XR(2); Go to TEST
TEST:	TXL	START,2,999	If XR(2) ≤ 999, Go to START
	HPR		Halt

- I/O PROCESSING
  - CPU initiates I/O by sending to IOP:
    - I/O Instruc.
      1. Device #
      2. IOP #
      3. Direction (I/O)
      4. Address of 1st instr. of I/O program in M.
  - IOP executes I/O program & interrupts CPU
  - IOP & CPU share memory (IOP "steals" cycles from CPU)

IBM 7094
(2nd Generation 704/709)

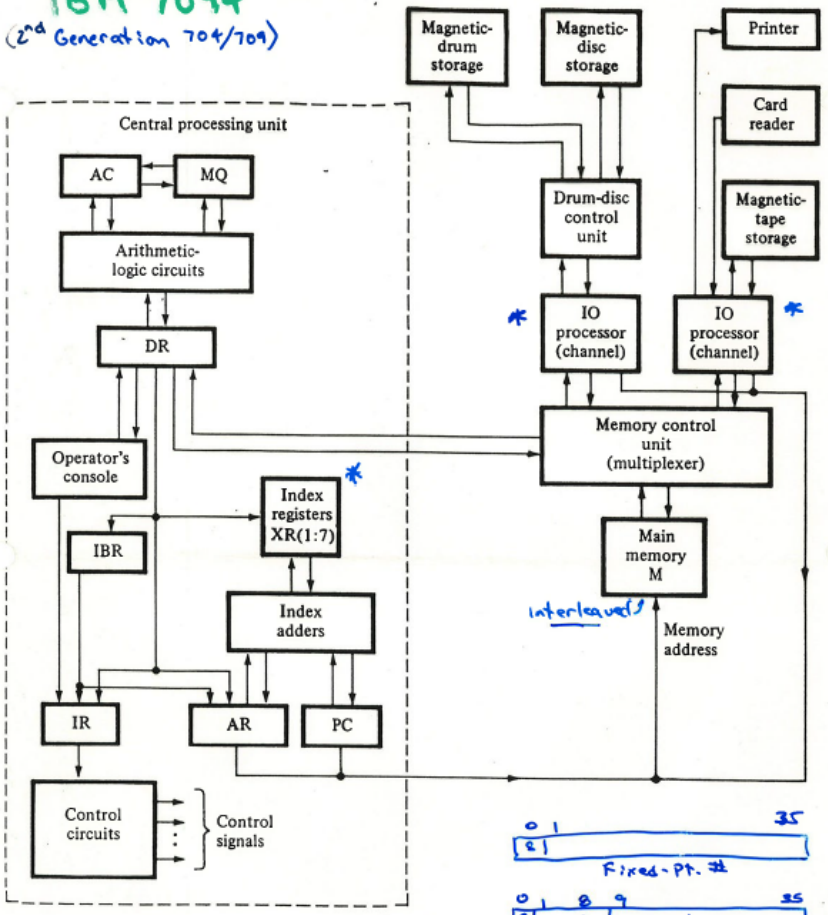


FIGURE 1.20
Structure of a second-generation computer: the IBM 7094.

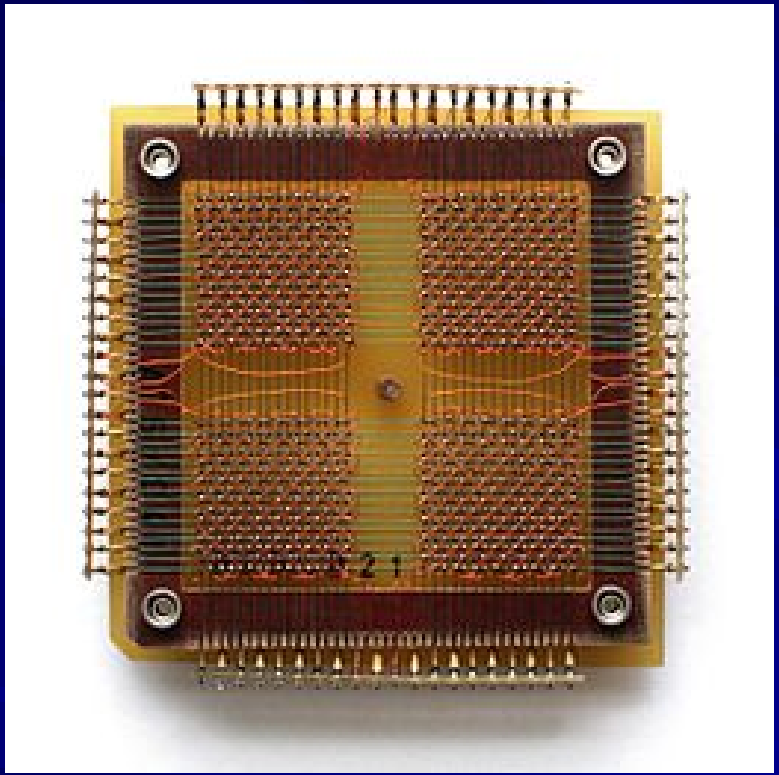
200 Instructions

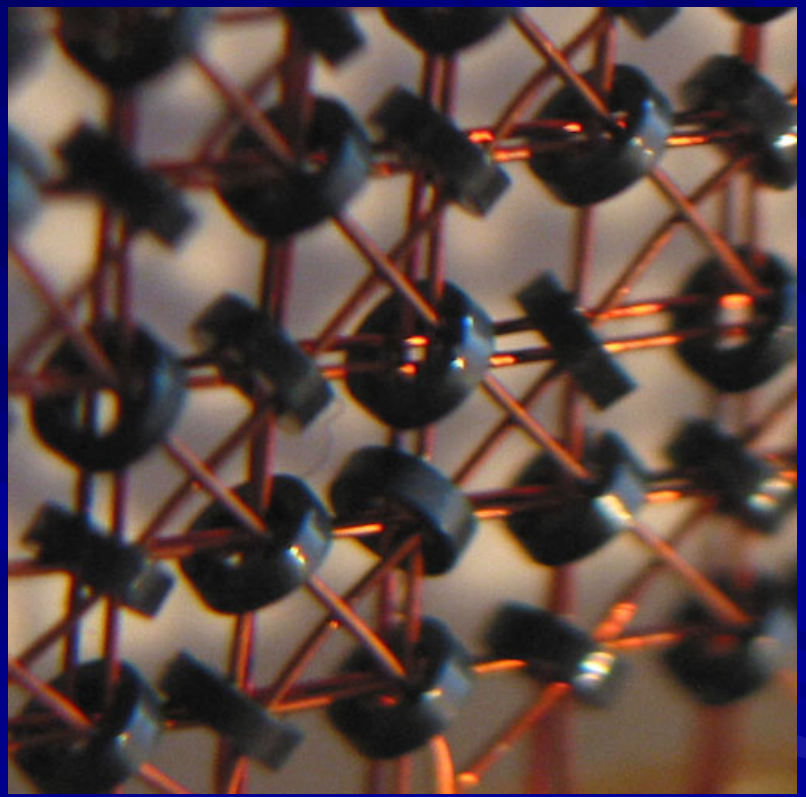
STORAGE FORMATS

1st & 2nd Generation UNIVAC

- UNIVAC I (1951 for Census Bureau)
  - Magnetic tape + error checking
  - 12-digit decimal arithmetic
  - 1-address instruction format
  - Mercury delay-line memory
- UNIVAC II (1957)
  - Magnetic core memory
- UNIVAC 1103/1103A (1956 – Scientific)
  - 36-bit binary arithmetic
  - 2-address instruction format
  - Floating-point hardware
  - interrupts

Magnetic core memory



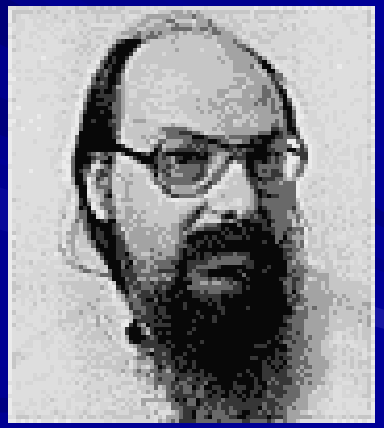


Third Generation Computers

- Beyond 1965
- Integrated circuit (IC) technology
- Semiconductor memories
- Memory hierarchy, virtual memories and caches
- Time-sharing
- Parallel processing and pipelining
- Microprogramming
- Examples: IBM 360 and 370, CYBER, ILLIAC IV, DEC PDP and VAX, Amdahl 470

C Programming Language and UNIX Operating System

1972



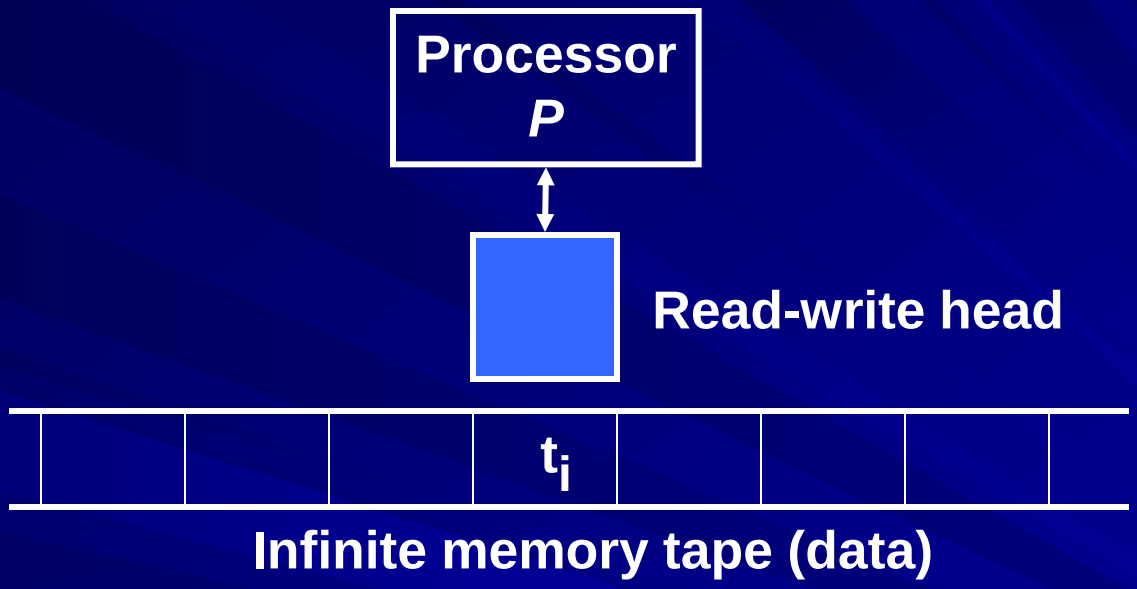
Now

Theory of Computing

- Alan Turing (1912-1954) gave a model of computing in 1936 – *Turing Machine*.
- Original paper: A. M. Turing, "On Computable Numbers with an Application to the *Entscheidungsproblem**," *Proc. Royal Math. Soc*., ser. 2, vol. 42, pp. 230-265, 1936.
- Recent book: David Leavitt, *The Man Who Knew Too Much: Alan Turing and the Invention of the Computer (Great Discoveries),* W. W. Norton & Co., 2005.

** The question of decidability, posed by mathematician Hilbert.*

Turing Machine



Machine instruction: s_h t_i o_j s_k

Present state of processor
Symbol on tape
Operation
Next state of processor

Turing Machine

- Four operations:
  - $o_j = t_j$, replace present symbol t_i by t_j
  - o_j = R, move head one position to right
  - o_j = L, move head one position to left
  - o_j = H, halt the computation
- Universal Turing Machine: small instruction set, #symbols × #states < 30; can perform any possible (*computable*) computation.
- *Computable* means that Turing machine halts in finite number of steps.
- Real computers have finite memory – they find certain problems *intractable*.

Ref: **J. P. Hayes, *Computer Architecture and Organization*, New York: McGraw-Hill, 1978.**

Example

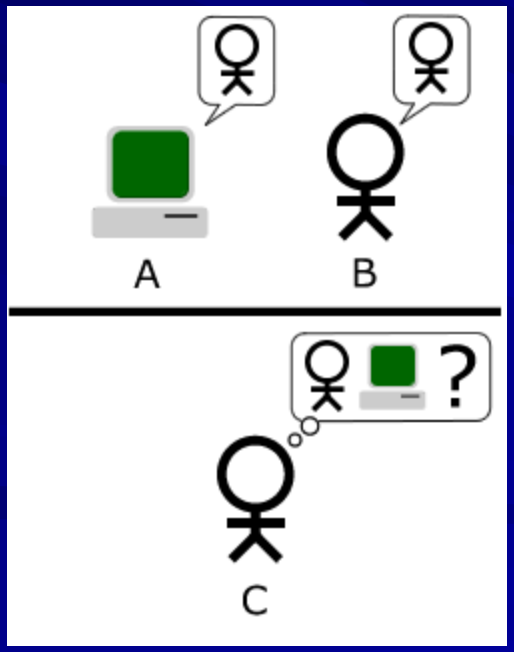
- Start with a blank tape and create a pattern 0b1b0b1b . . .
- Define symbols: b (blank), 0, 1

Present state	Symbol on tape	Operation	Next state
S0 (begin)	blank	Write 0 and move right	S1
S1	blank	Move right	S2
S2	blank	Write 1 and move right	S3
S3	blank	Move right	S0

http://en.wikipedia.org/wiki/Turing_machine_examples

Turing Test

- Can a computer think? (Turing, 1950).
- http://en.wikipedia.org/wiki/Computing_Machinery_and_Intelligence#cite_note-1
- A. P. Saygin, I. Cicekli and V. Akman, “Turing Test: 50 Years Later,” *Minds and Machines*, vol. 10, no. 4, pp. 463-518, 2000.



Watson vs. humans: http://www.engadget.com/2011/01/13/ibms-watson-supercomputer-destroys-all-humans-in-jeopardy-pract/

Incompleteness Theorem

- In 1931, the Czech-born mathematician Kurt Gödel (1906-1978) demonstrated that within any given branch of mathematics, there would always be some propositions that couldn't be proven either true or false using the rules and axioms.
- Gödel's Theorem has been used to argue that a computer can never be as smart as a human being because the extent of its knowledge is limited by a fixed set of axioms, whereas people can discover unexpected truths.
- See http://www.miskatonic.org/godel.html and other websites.

The Barber Paradox
By Linda Shaver
(An Example of Undecidability)

In a particular town, there's a particular barbershop with a peculiar sign in the window that reads:

"This barber shaves all and only those men of the town who do not shave themselves."

Question: According to the sign in the window, does the barber shave himself?

The Now Generation

- Personal computers
- Laptops and Palmtops
- Networking and wireless
- SOC and MEMS technology
- And the future!
  - Biological computing
  - Molecular computing
  - Nanotechnology
  - Optical computing
  - Quantum computing
  - See articles listed on the next slide and available at E7700: Advanced VLSI Design course site, http://www.eng.auburn.edu/~vagrawal/COURSE/E7770_Spr12/course.html

Useful Reading

- R. I. Bahar, D. Hammerstrom, J. Harlow, W. H. Joyner Jr., C. Lau, D. Marculescu, A. Orailoglu and M. Pedram, “Architectures for Silicon Nanoelectronics and Beyond,” *Computer*, vol. 40, no. 1, pp. 25-33, January 2007.
- T. Munakata, “Beyond Silicon: New Computing Paradigms, “*Comm. ACM*, vol. 50, no. 9, pp. 30-34, Sept. 2007.
- W. Robinett, G. S. Snider, P. J. Kuekes and S. Williams, “Computing with a Trillion Crummy Components,” *Comm. ACM*, vol. 50, no. 9, pp. 35-39, Sept. 2007.
- J. Kong, “Computation with Carbon Nanotube Devices,” *Comm. ACM*, vol. 50, no. 9, pp. 40-42, Sept. 2007.
- R. Stadler, “Molecular, Chemical and Organic Computing,” *Comm. ACM*, vol. 50, no. 9, pp. 43-45, Sept. 2007.
- M. T. Bohr, R. S. Chau, T. Ghani and K. Mistry, "The High-k Solution," *IEEE Spectrum*, vol. 44, no. 10, pp. 29-35, October 2007.
- J. H. Reif and T. H. Labean, “Autonomous Programmable Biomolecular Devices using Self-Assembled DNA Nanostructures,” *Comm. ACM*, vol. 50, no. 9, pp. 46-53, Sept. 2007.
- D. Bacon and D. Leung, “Toward a World with Quantum Computers,” *Comm. ACM*, vol. 50, no. 9, pp. 55-59, Sept. 2007.
- H. Abdeldayem and D. A. Frazier, “Optical Computing: Need and Challenge,” *Comm. ACM*, vol. 50, no. 9, pp. 60-62, Sept. 2007.
- D. W. M. Marr and T. Munakata, “Micro/Nanofluidic Computing,” *Comm. ACM*, vol. 50, no. 9, pp. 64-68, Sept. 2007.
- M. Aono, M. Hara and K. Aihara, “Amoeba-Based Neurocomputing with Chaotic Dynamics,” *Comm. ACM*, vol. 50, no. 9, pp. 69-72, Sept. 2007.

Next: The MIPS ISA

- MIPS (*Microprocessor without Interlocked Pipeline Stages*) is a reduced instruction set architecture.