Definition(s) of system

- A system can be broadly defined as an integrated set of elements that accomplish a defined objective.
- People from different engineering disciplines have different perspectives of what a "system" is.
- For example,
  - software engineers often refer to an integrated set of computer programs as a "system"
  - electrical engineers might refer to complex integrated circuits or an integrated set of electrical units as a "system"
- As can be seen, "system" depends on one's perspective, and the "integrated set of elements that accomplish a defined objective" is an appropriate definition.

Definition(s) of system

- A system is an assembly of parts where:
  - The parts or components are connected together in an organized way.
  - The parts or components are affected by being in the system (and are changed by leaving it).
  - The assembly does something.
  - The assembly has been identified by a person as being of special interest.
- Any arrangement which involves the handling, processing or manipulation of resources of whatever type can be represented as a system.

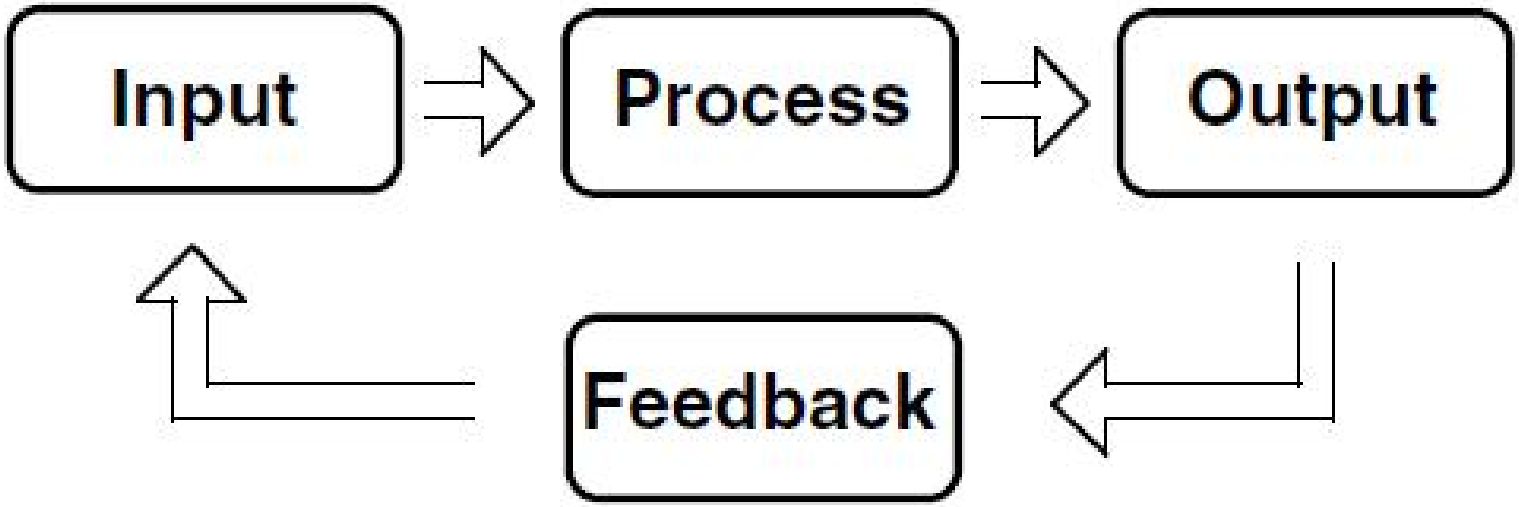
Definition(s) of system

- A system is defined as multiple parts working together for a common purpose/goal.
- Systems can be large and complex
  - such as the air traffic control system or our global telecommunication network.
- Small devices can also be considered as systems
  - such as a pocket calculator, alarm clock, or 10-speed bicycle.

Definition(s) of system

- Systems have inputs, processes, and outputs.
- When feedback (direct or indirect) is involved, that component is also important to the operation of the system.
- To explain all this, systems are usually explained using a model.
- A model helps to illustrate the major elements and their relationship, as illustrated in the next slide

A systems model



Information Systems

- The ways that organizations
  - Store
  - Move
  - Organize
  - Process

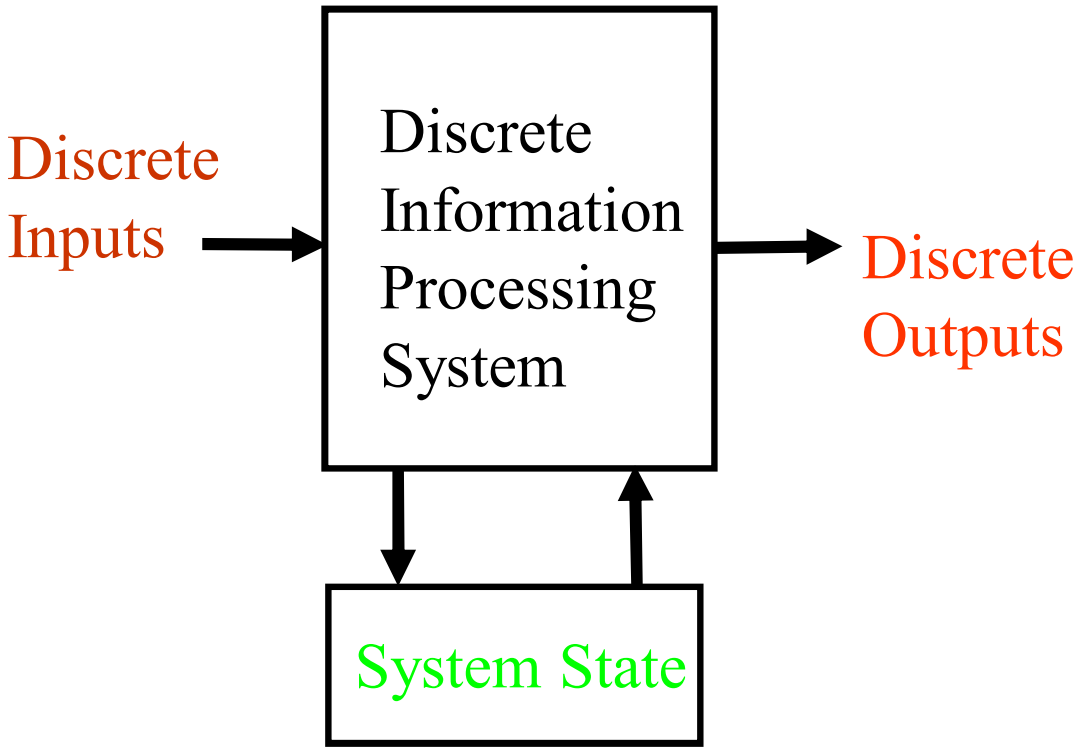
  their information

Information Technology

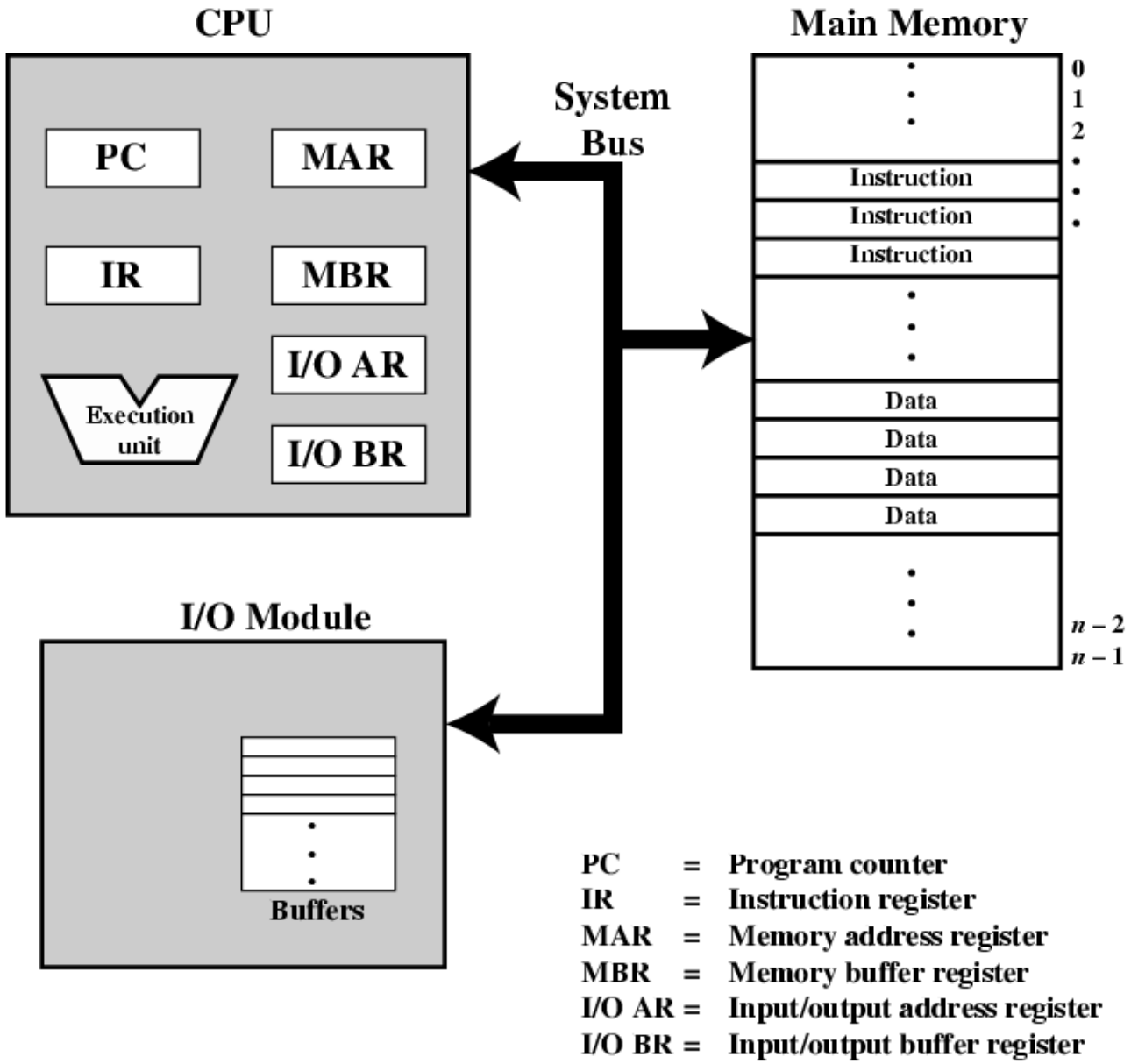
- Components that implement information systems,
  - Hardware
    - physical tools: computer and network hardware, but also low-tech things like pens and paper
  - Software
    - (changeable) instructions for the hardware
  - People
  - Procedures
    - instructions for the people
  - Data/databases

Digital System

- **Takes a set of discrete information (inputs) and discrete internal information (system state) and generates a set of discrete information (outputs).**



A Digital Computer:Top Level View



A Digital Computer Example

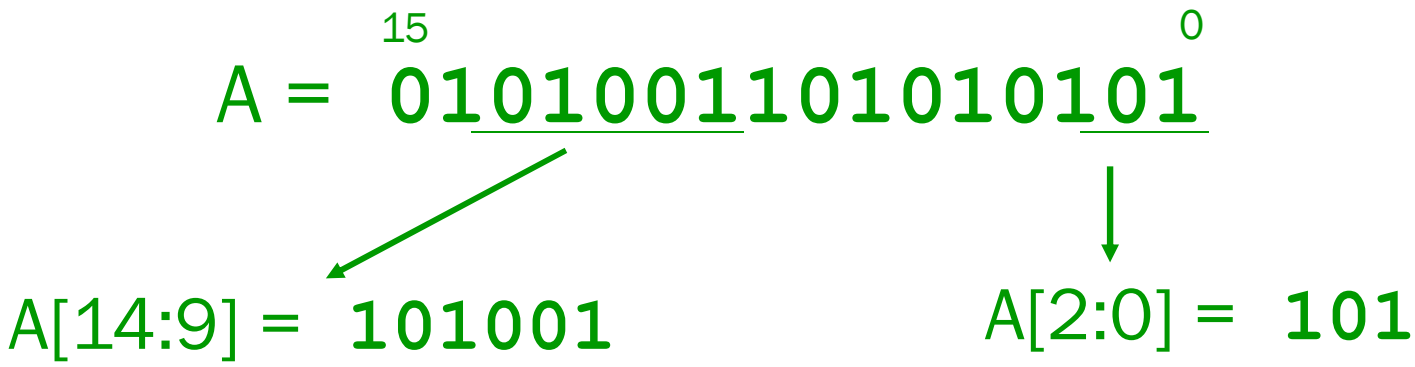
Inputs:
Keyboard,
mouse, modem,
microphone

Outputs: CRT,
LCD, modem,
speakers

Representing Multi-bit Values

- Number bits from right (0) to left (n-1)
  - just a convention -- could be left to right, but must be *consistent*
- Use brackets to denote range:

D[l:r] denotes bit **l** to bit **r**, from *left* to *right*



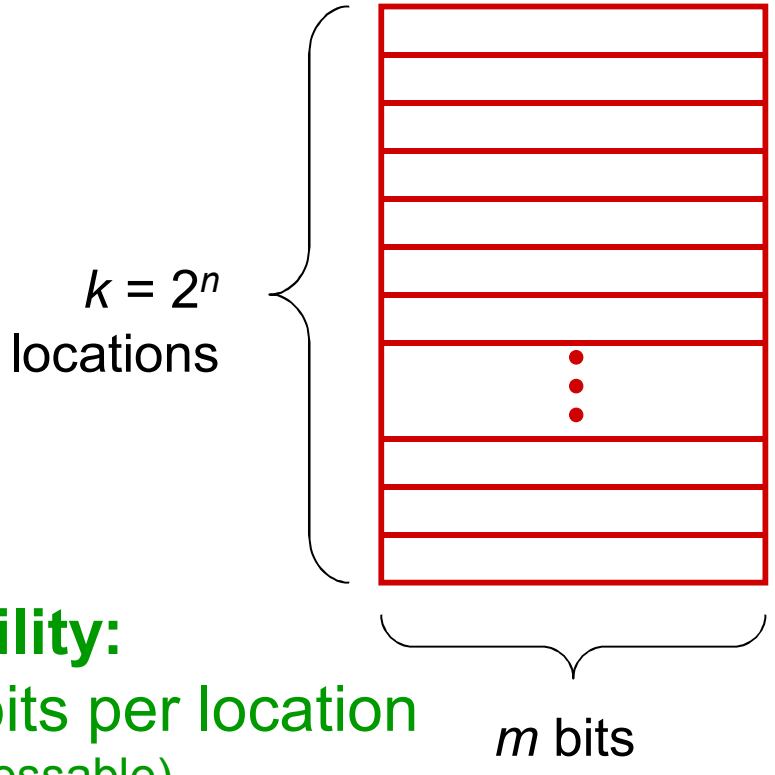
- May also see A**<14:9>**,

especially in hardware block diagrams.

Memory

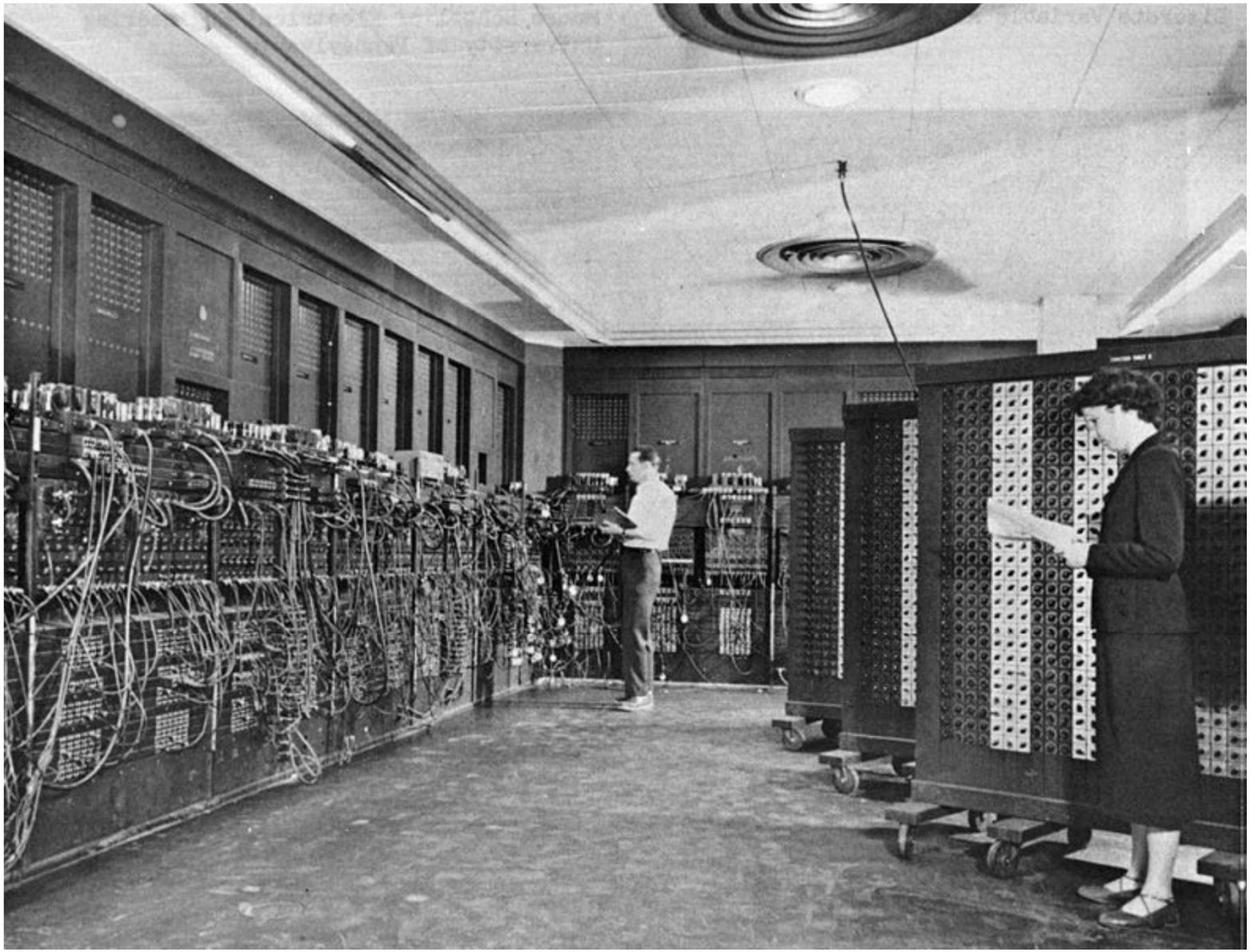
•Now that we know how to store bits, we can build a memory – a logical $k \times m$ array of stored bits.

Address Space:
number of locations
(usually a power of 2)



Addressability:
number of bits per location
(e.g., byte-addressable)

ENIAC



ENIAC – background -- 1G

- Electronic Numerical Integrator And Computer
- Eckert and Mauchly designed and built ENIAC (1943-45) at the University of Pennsylvania
- The first, completely electronic, operational, general-purpose analytical calculator!
  - —30 tons, 72 square meters, 200KW
- Performance
  - —Read in 120 cards per minute
  - —Addition took 200 μs, Division 6 ms
- Not very reliable!
- Trajectory tables for weapons
- Started 1943, Finished 1945; Used until 1955
  - —Too late for war effort

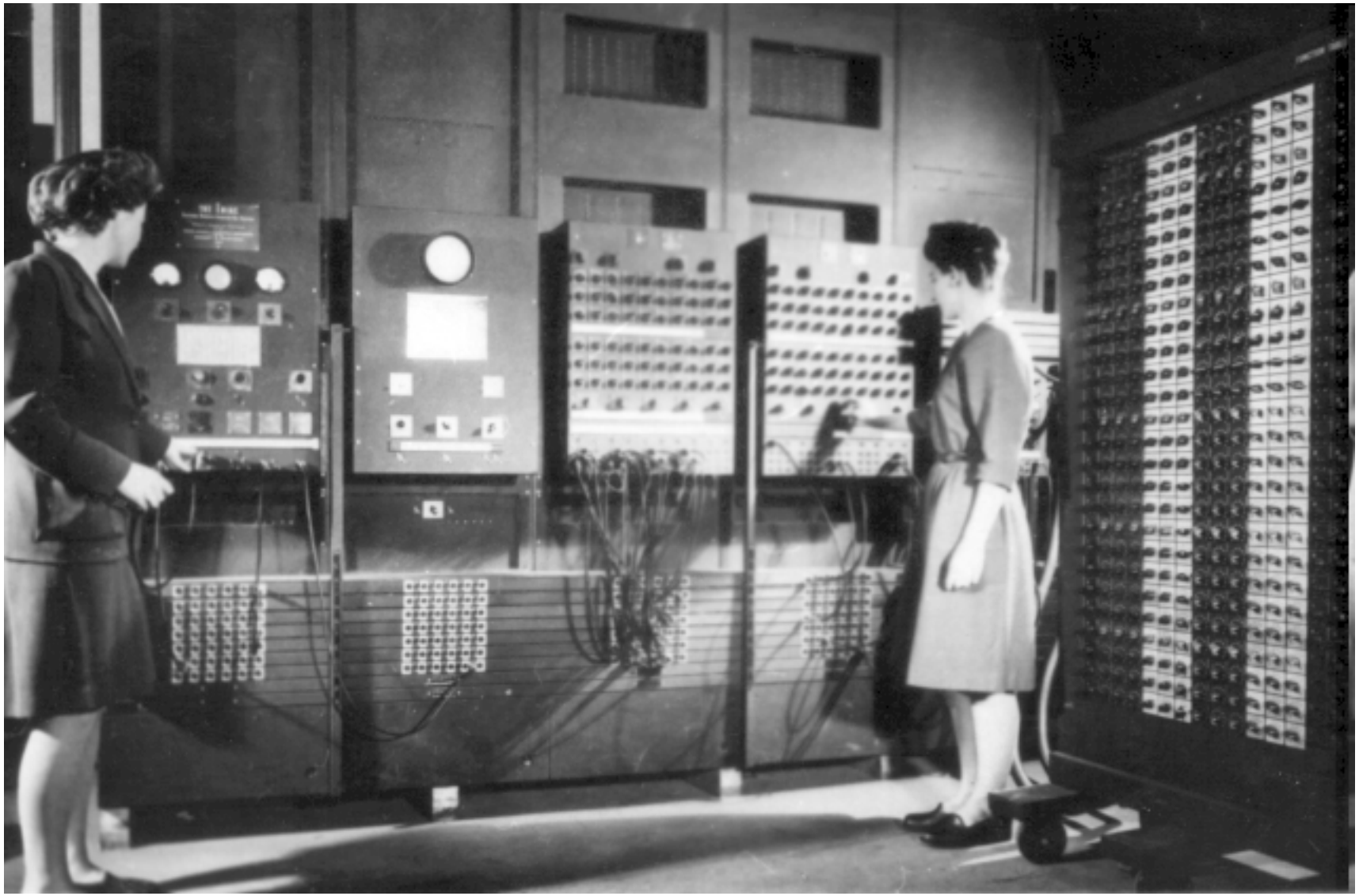
Computer Architecture: A Little History

Sometimes we'll use a historical event to help understand why certain ideas arose

Why worry about old ideas?

- Helps to illustrate the design process, and explains why certain decisions were taken
- Because future technologies might be as constrained as older ones
- Those who ignore history are doomed to repeat it
  - —Every mistake made in mainframe design was also made in minicomputers, then microcomputers, where next?

Programming in those days

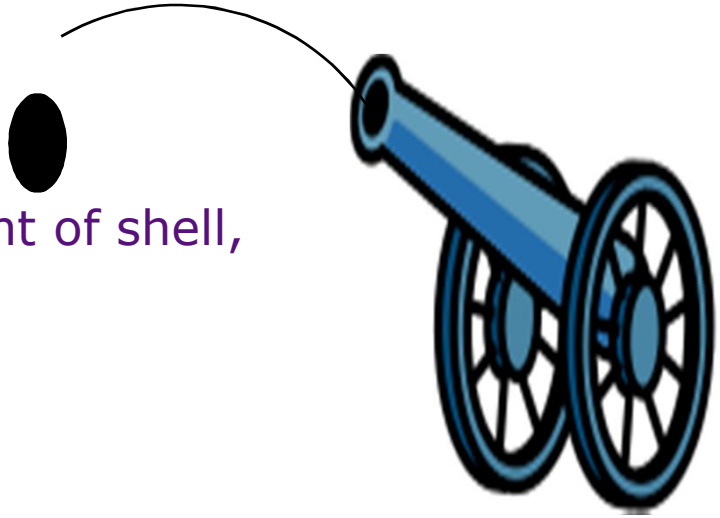


Electronic Numerical Integrator and Computer (ENIAC)

WW-2 Effort

Application: Ballistic calculations

angle = f (location, tail wind, cross wind,
air density, temperature, weight of shell,
propellant charge, ...)



ENIAC - details

- Decimal (not binary)
- 20 accumulators of 10 digits
  - Each accumulator had 100 vacuum tubes
  - One decimal digit was represented by a ring of 10 tubes; only one "on" at a time
- Programmed manually by switches
  - Plug / unplug cables
- 18,000 vacuum tubes
- 30 tons
- 15,000 square feet
- 140 kW power consumption
- 5,000 additions per second

programming

- Too time consuming
- ENIAC's programming system was external
  - Sequences of instructions were executed independently of the results of the calculation
  - Human intervention required to take instructions "out of order"
- Eckert, Mauchly, John von Neumann and others designed to solve this problem
  - Solution was the *stored program computer*

    $\Rightarrow$ *"program can be manipulated as data"*
  - both program and data were in memory
  - Gets instruction from memory to execute
  - Change program by setting portion of memory

Stored Program Computer

Program = A sequence of instructions

How to control instruction sequencing?

manual control	calculators	
automatic control		
internal		
plug board	ENIAC	1946
read-only memory	ENIAC	1948
read-write memory	EDVAC	1947 *(concept)*

- The same storage can be used to store **program and data**

von Neumann/Turing

- **Stored Program** concept
- Features
  - Both data and instructions (programs) are stored in a single read/write memory
  - Memory contents are addressable by location, regardless of the content itself
  - Sequential execution
  - ALU operating on binary data
  - Control unit interpreting instructions from memory and executing
  - Input and output equipment operated by control unit
- Princeton **I**nstitute for **A**dvanced **S**tudies
  - **IAS** computer
- Completed 1952

Paraphrased from von Neumann's paper:

- 1. Since the device is primarily a computer it will have to perform the elementary **operations** of arithmetic most frequently. Therefore, it should contain specialized organs for just these operations, i.e., addition, subtraction, multiplication, and division.
- 2. The logical control of the device (i.e., proper sequencing of its operations) can best be carried out by a central **control** organ.
- 3. A device which is to carry out long and complicated sequences of operation must have a considerable **memory** capacity.

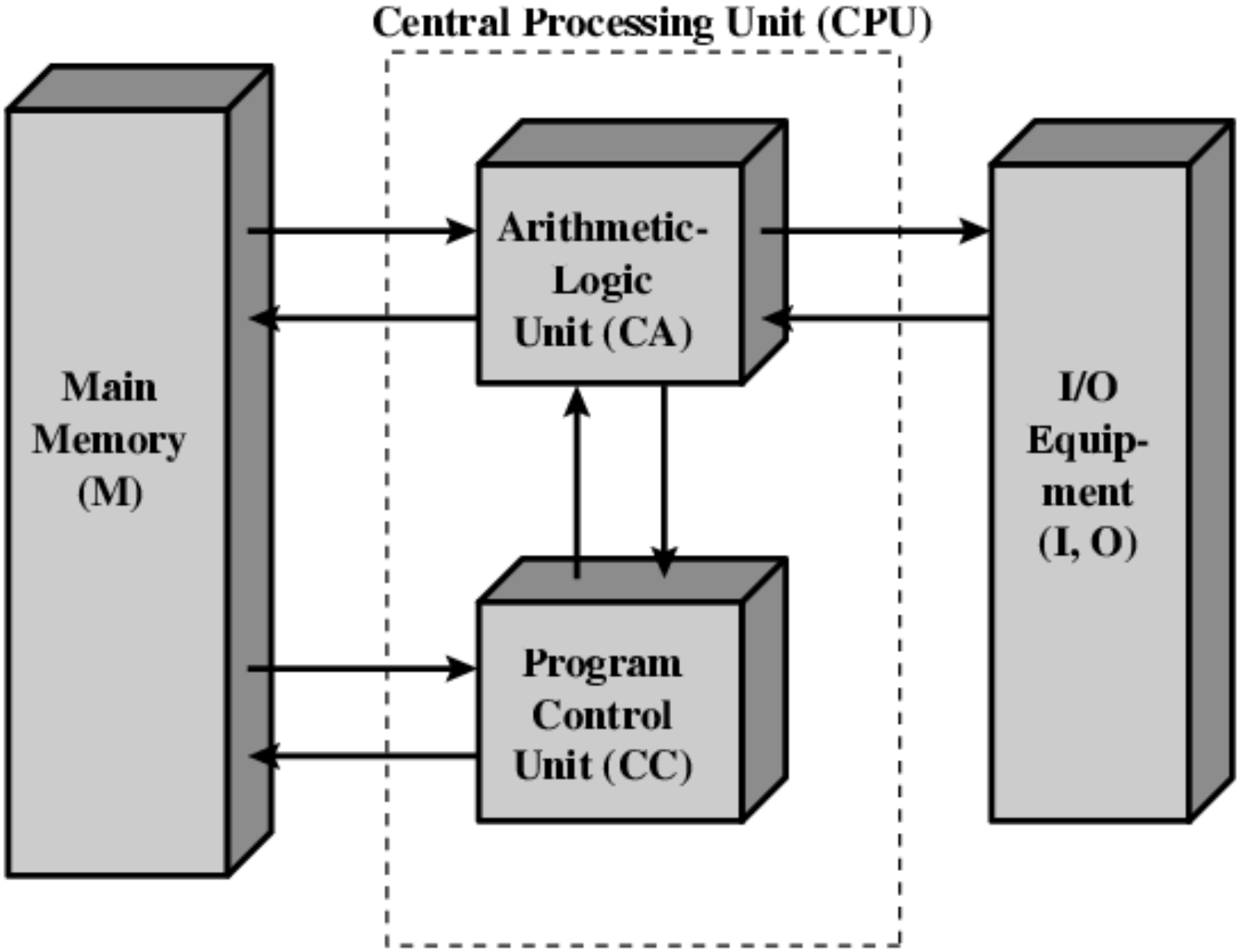
Von Neumann (cont'd)

- 4. The device must have organs to transfer information from the outside recording medium of the device into the central arithmetic part and central control part, and the memory. These organs form its **input**.
- 5. The device must have organs to transfer information from the central arithmetic part and central control part, and the memory into the outside recording medium. These organs form its **output**.

IAS Machine

- This architecture (just described) came to be known as the “von Neumann” architecture and has been the basis for virtually every machine designed since then

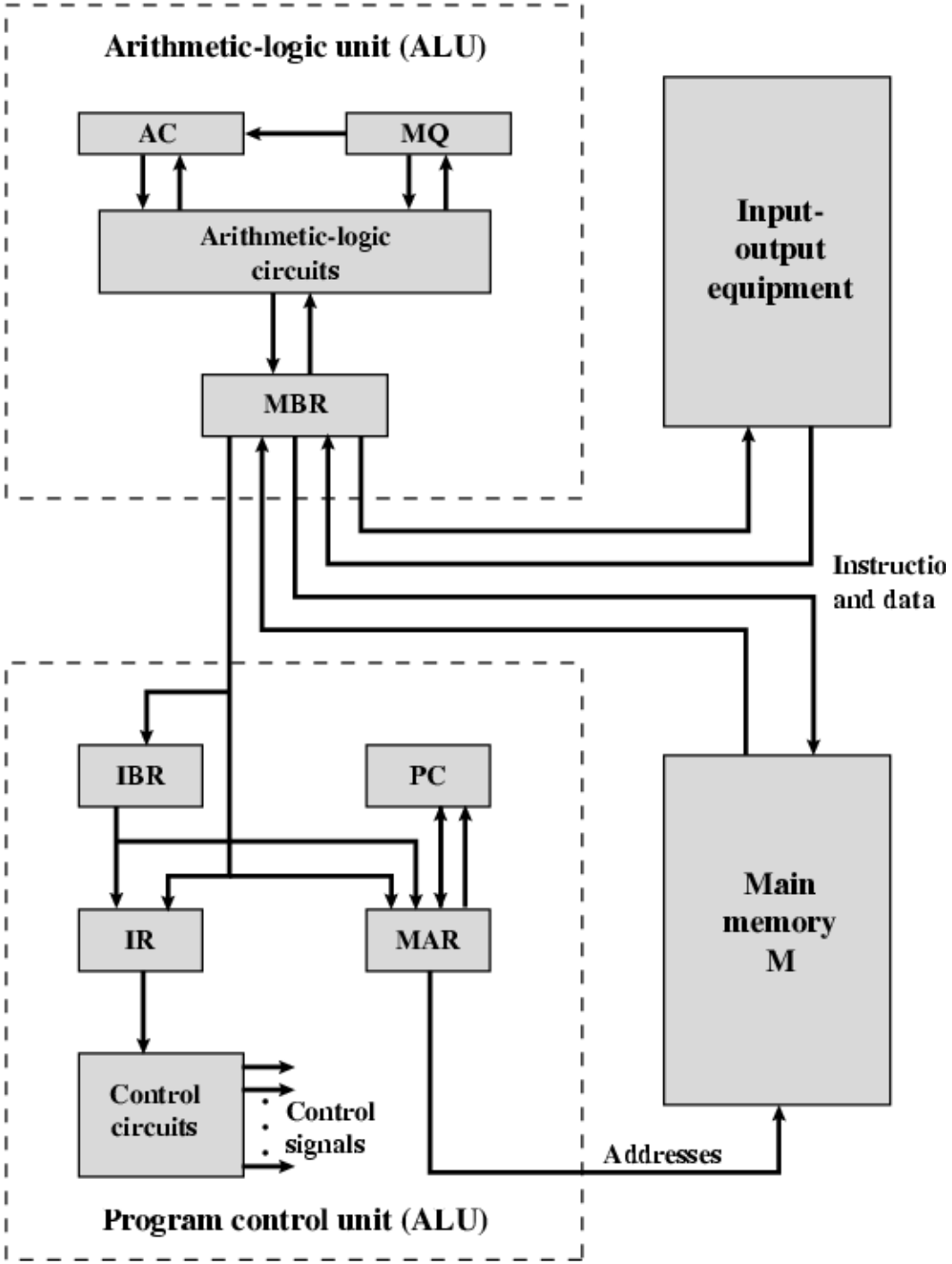
Structure of von Neumann machine



IAS - details

- 1000 x 40 bit words
  - Binary number
  - 2 x 20 bit instructions
- Set of registers (storage in CPU)
  - Memory Buffer Register
  - Memory Address Register
  - Instruction Register
  - Instruction Buffer Register
  - Program Counter
  - Accumulator
  - Multiplier Quotient

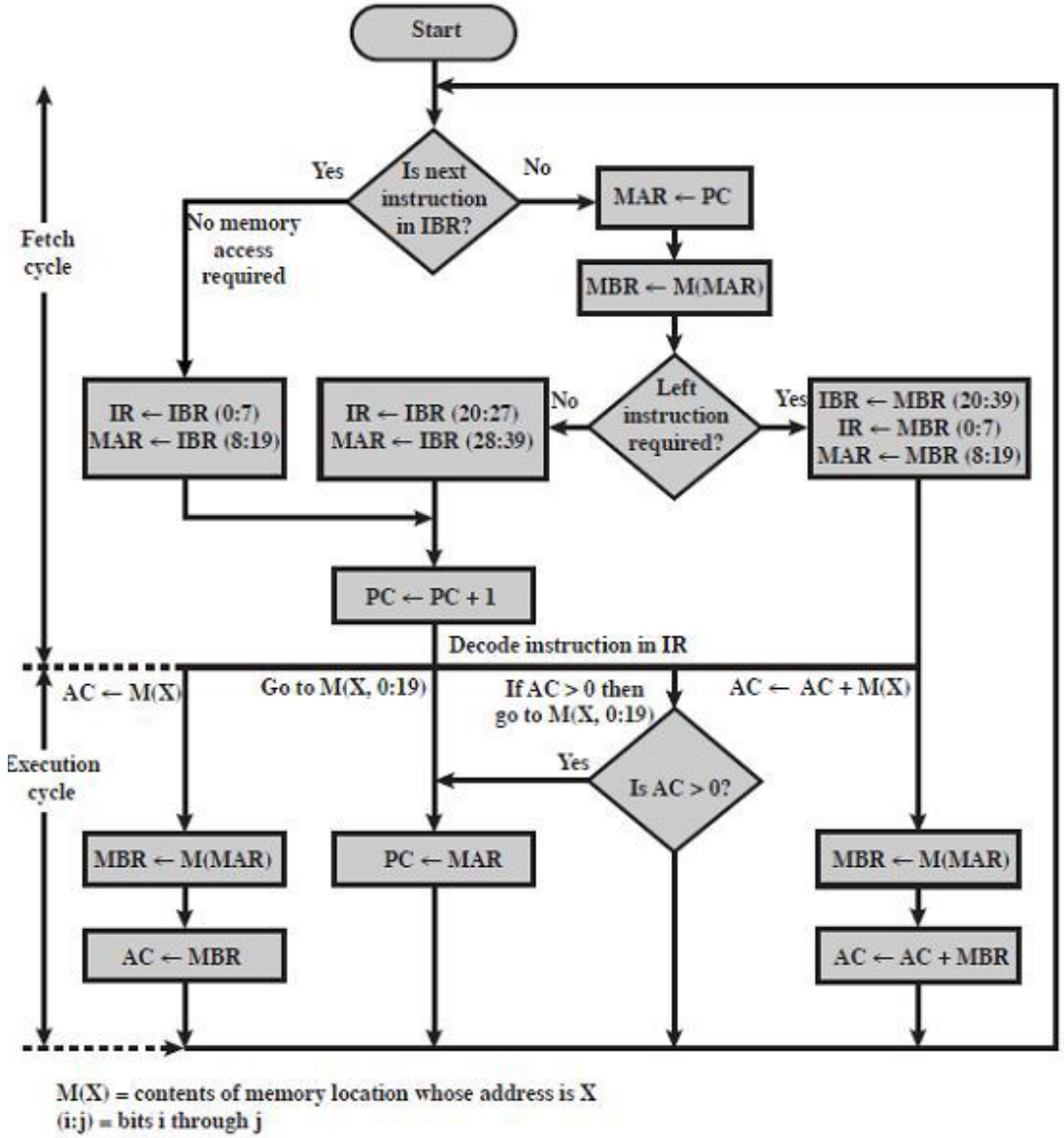
Structure of IAS – detail



IAS Instruction Set

Instruction Type	Opcode	Representation	Description
Data transfer	00001010	LOAD MQ	Transfer contents of register MQ to the accumulator AC
	00001001	LOAD MQ,M(X)	Transfer contents of memory location X to MQ
	00100001	STOR M(X)	Transfer contents of accumulator to memory location X
	00000001	LOAD M(X)	Transfer M(X) to the accumulator
	00000010	LOAD –M(X)	Transfer –M(X) to the accumulator
	00000011	LOAD |M(X)|	Transfer absolute value of M(X) to the accumulator
	00000100	LOAD –|M(X)|	Transfer –|M(X)| to the accumulator
Unconditional branch	00001101	JUMP M(X,0:19)	Take next instruction from left half of M(X)
	00001110	JUMP M(X,20:39)	Take next instruction from right half of M(X)
Conditional branch	00001111	JUMP+ M(X,0:19)	If number in the accumulator is nonnegative, take next instruction from left half of M(X)
	00010000	JUMP+ M(X,20:39)	If number in the accumulator is nonnegative, take next instruction from right half of M(X)
Arithmetic	00000101	ADD M(X)	Add M(X) to AC; put the result in AC
	00000111	ADD |M(X)|	Add |M(X)| to AC; put the result in AC
	00000110	SUB M(X)	Subtract M(X) from AC; put the result in AC
	00001000	SUB |M(X)|	Subtract |M(X)| from AC; put the remainder in AC
	00001011	MUL M(X)	Multiply M(X) by MQ; put most significant bits of result in AC, put least significant bits in MQ
	00001100	DIV M(X)	Divide AC by M(X); put the quotient in MQ and the remainder in AC
	00010100	LSH	Multiply accumulator by 2, i.e., shift left one bit position
	00010101	RSH	Divide accumulator by 2, i.e., shift right one position
Address modify	00010010	STOR M(X,8:19)	Replace left address field at M(X) by 12 rightmost bits of AC
	00010011	STOR M(X,28:39)	Replace right address field at M(X) by 12 rightmost bits of AC

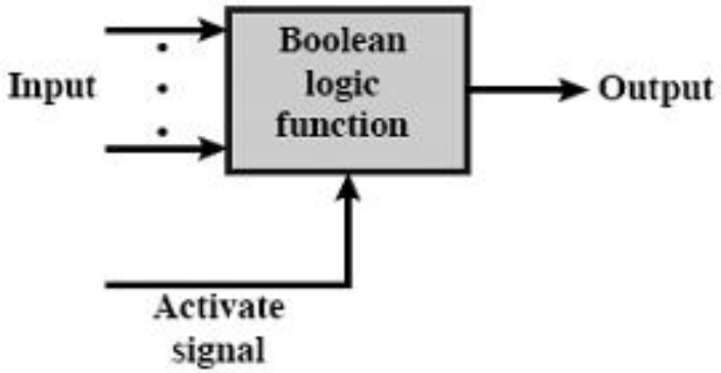
IAS Operation



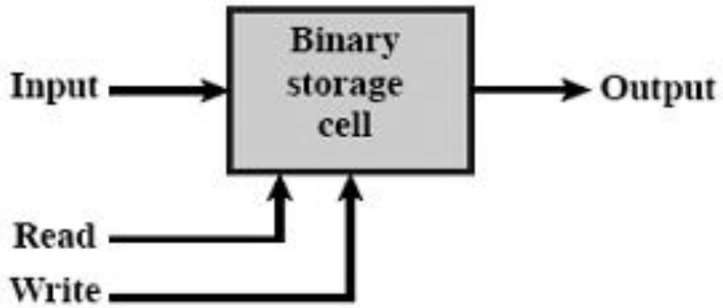
M(X) = contents of memory location whose address is X
(i:j) = bits i through j

Computer Components

- Only two basic components are needed to create a computer: **gates** and **memory cells**
- Each component is composed of multiple transistors



(a) Gate

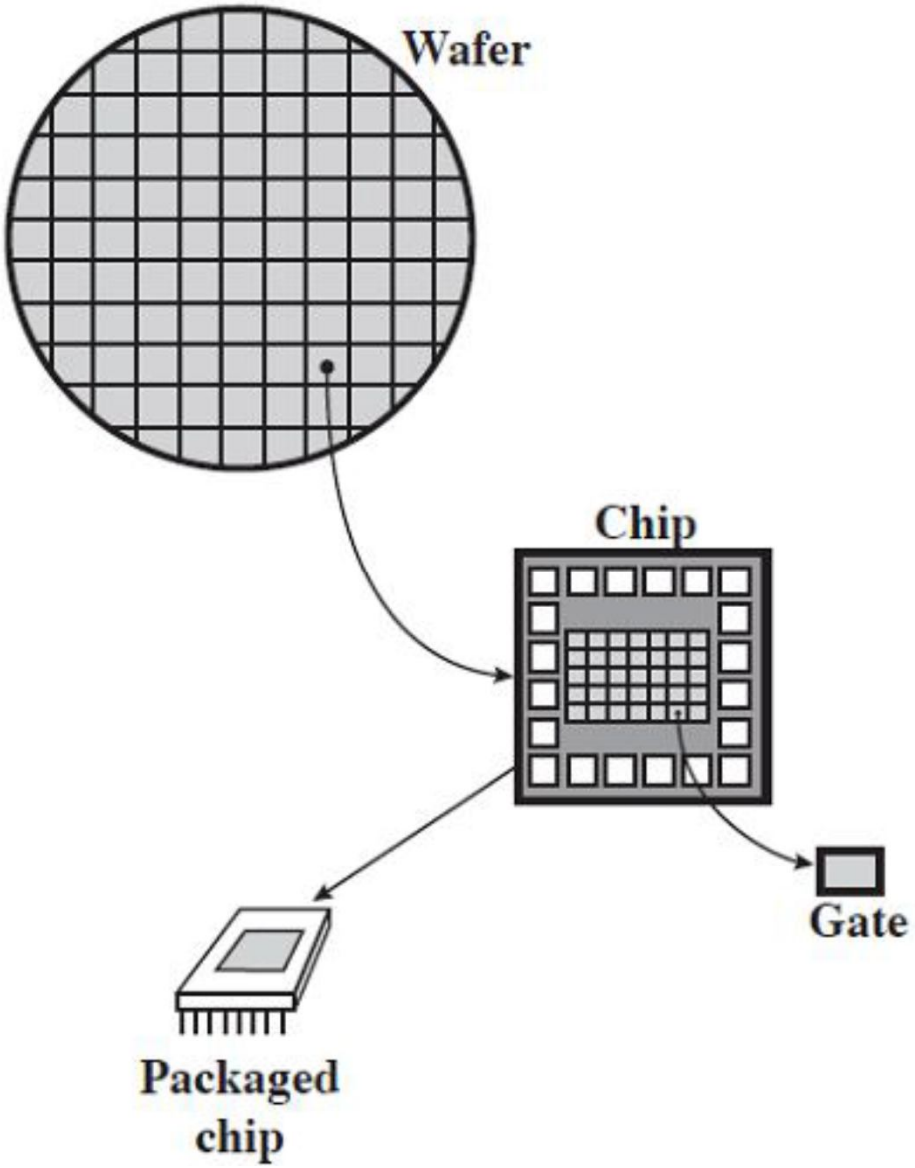


(b) Memory cell

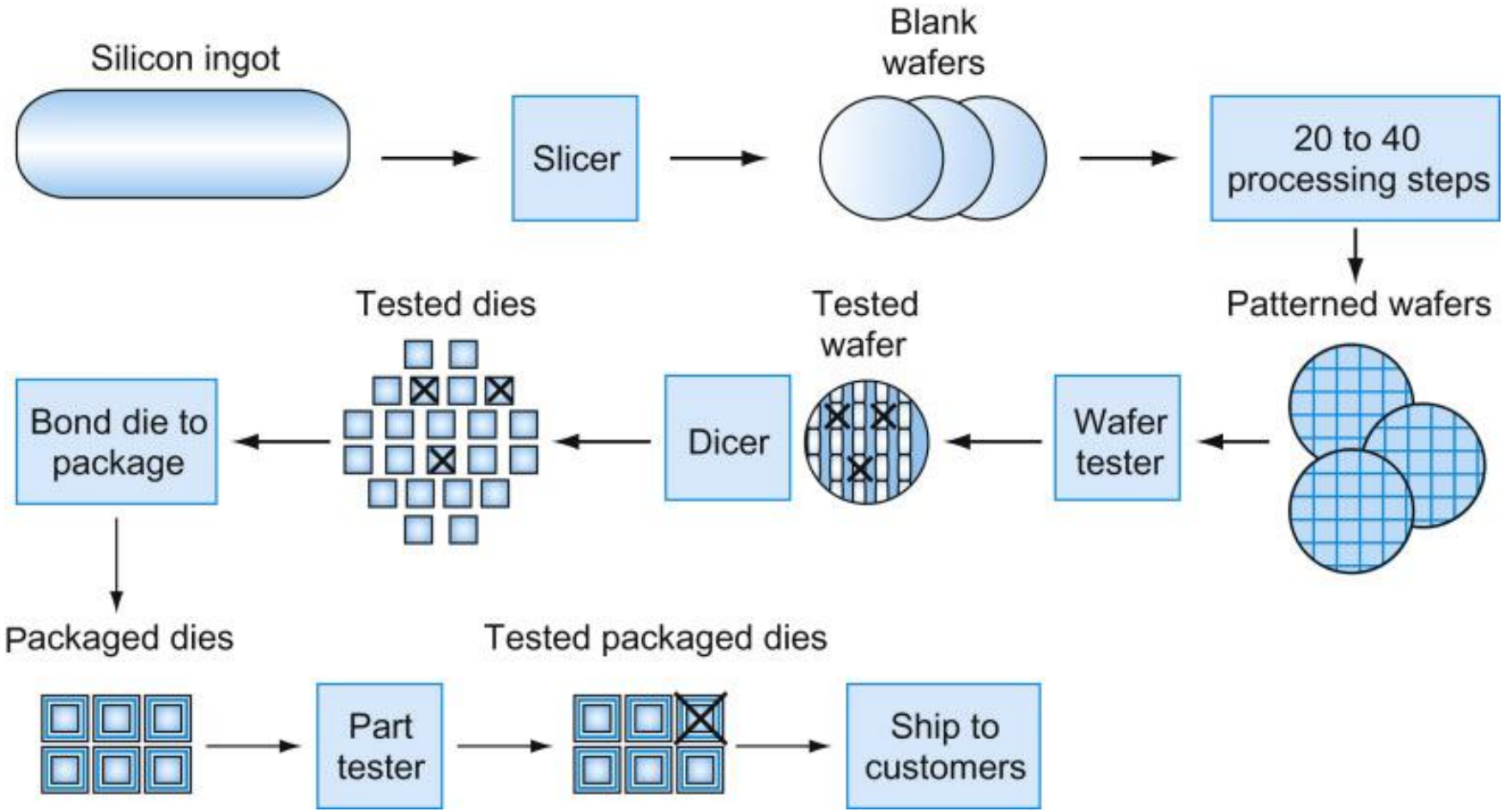
Four basic functions

- Data Storage: memory cells
- Data Processing: gates
- Data Movement: connections (paths) between gates and memory cells
- Control: control signals activate gates and memory and determine whether memory access is read or write
- Gates, memory cells and interconnections can be etched into a single **piece of silicon**

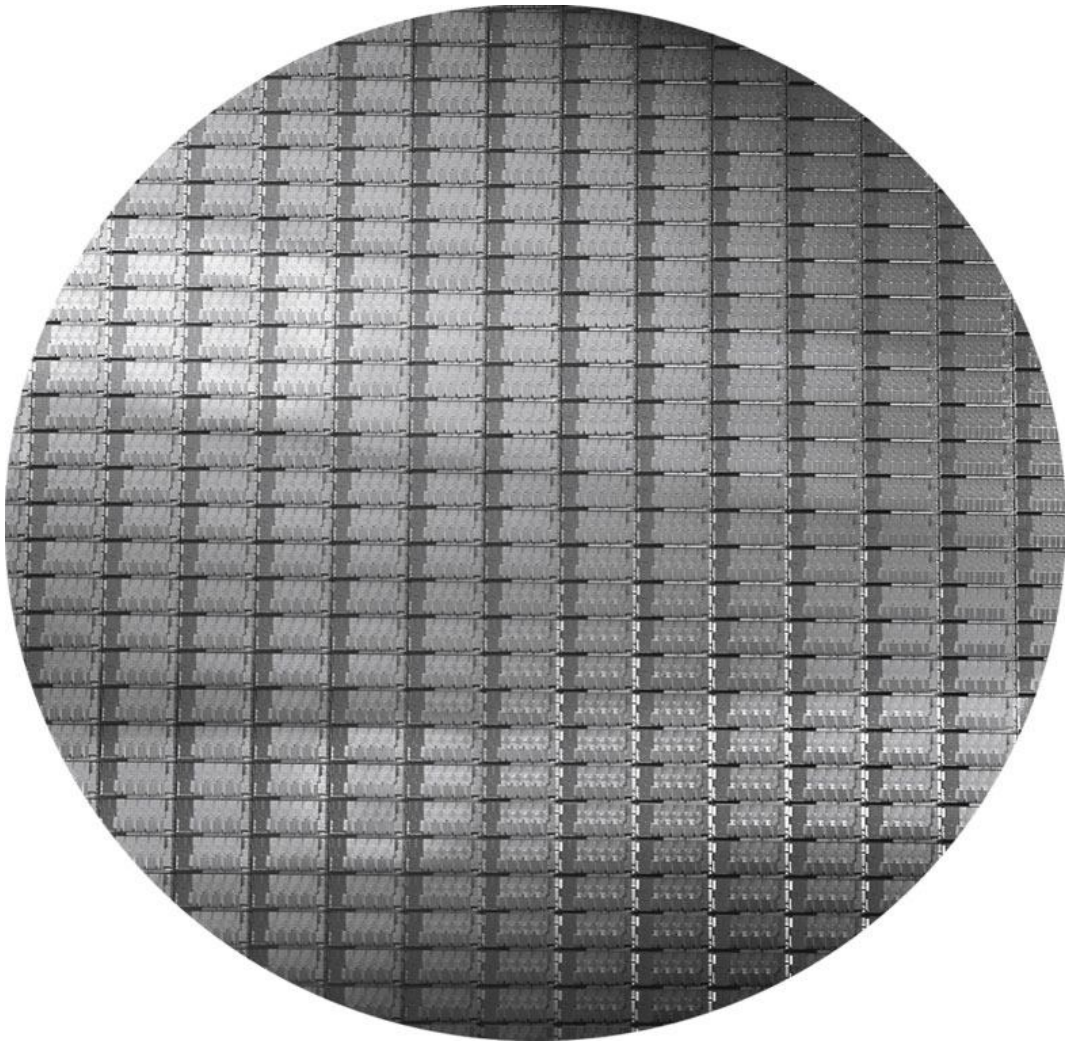
Chip manufacturing



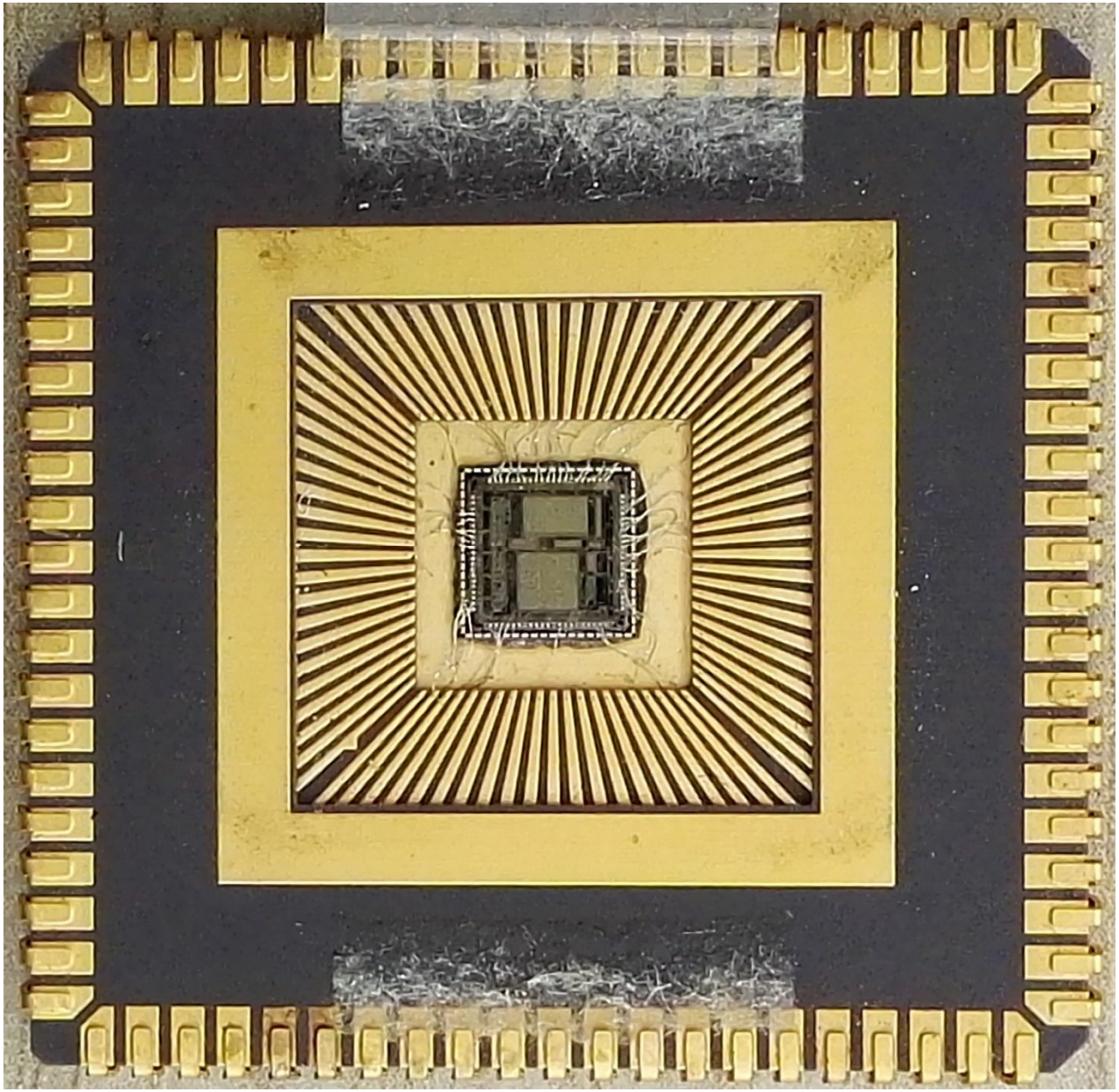
The chip manufacturing process



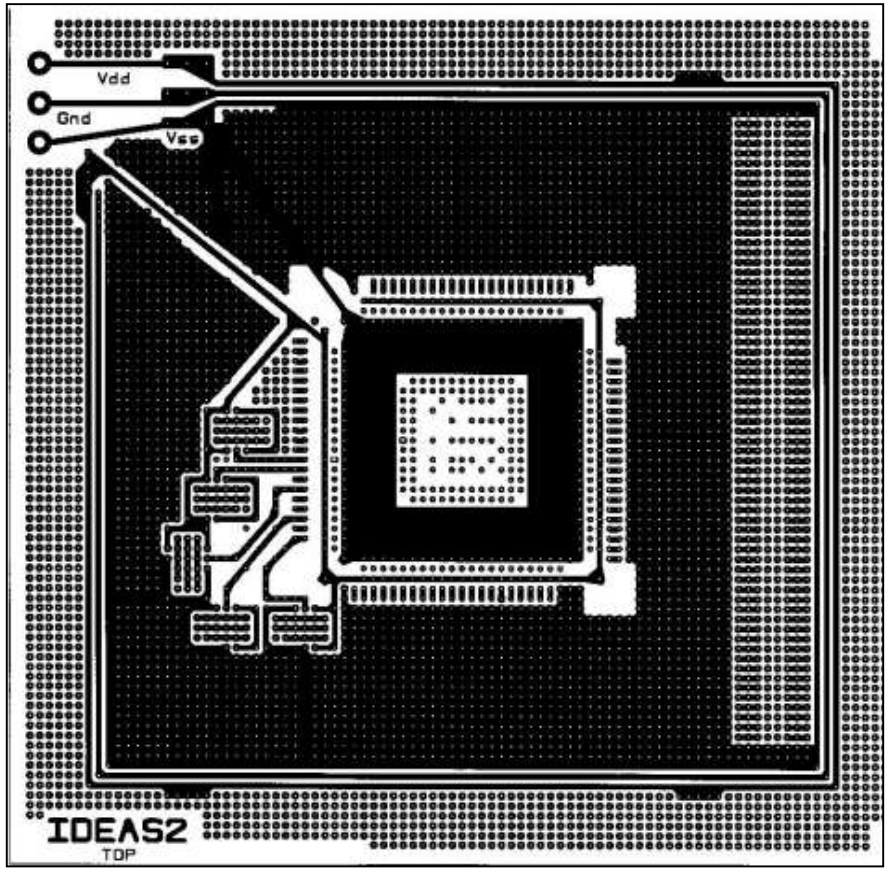
A 12-inch (300 mm) wafer of Intel Core i7 (Courtesy Intel)



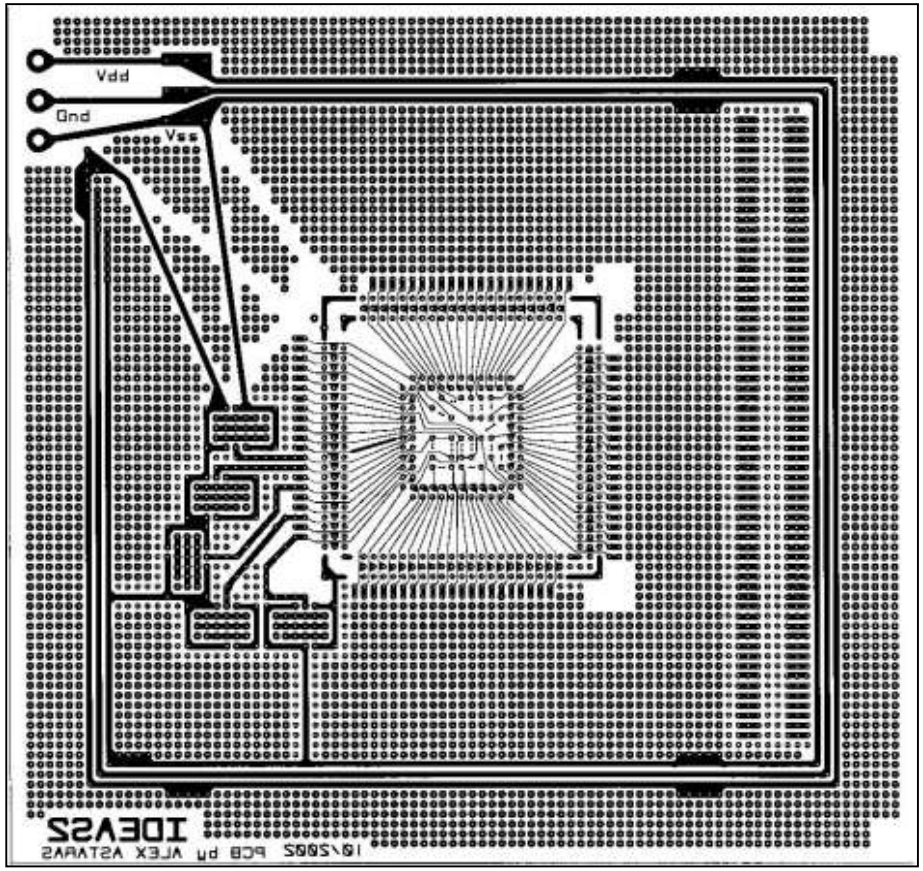
Picture of the fabricated chip



Testing board for the chip



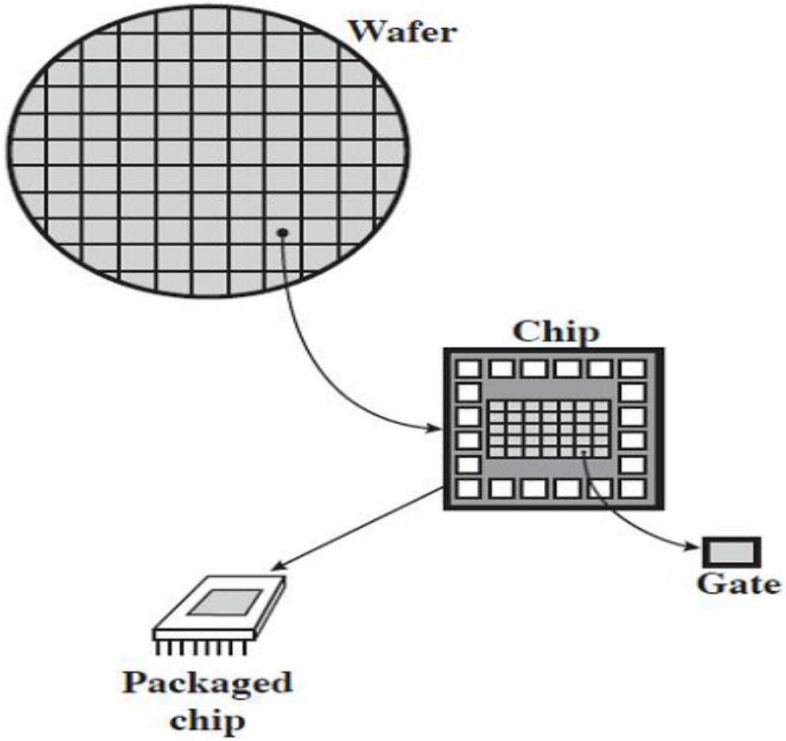
Top side



Bottom side

Microelectronics

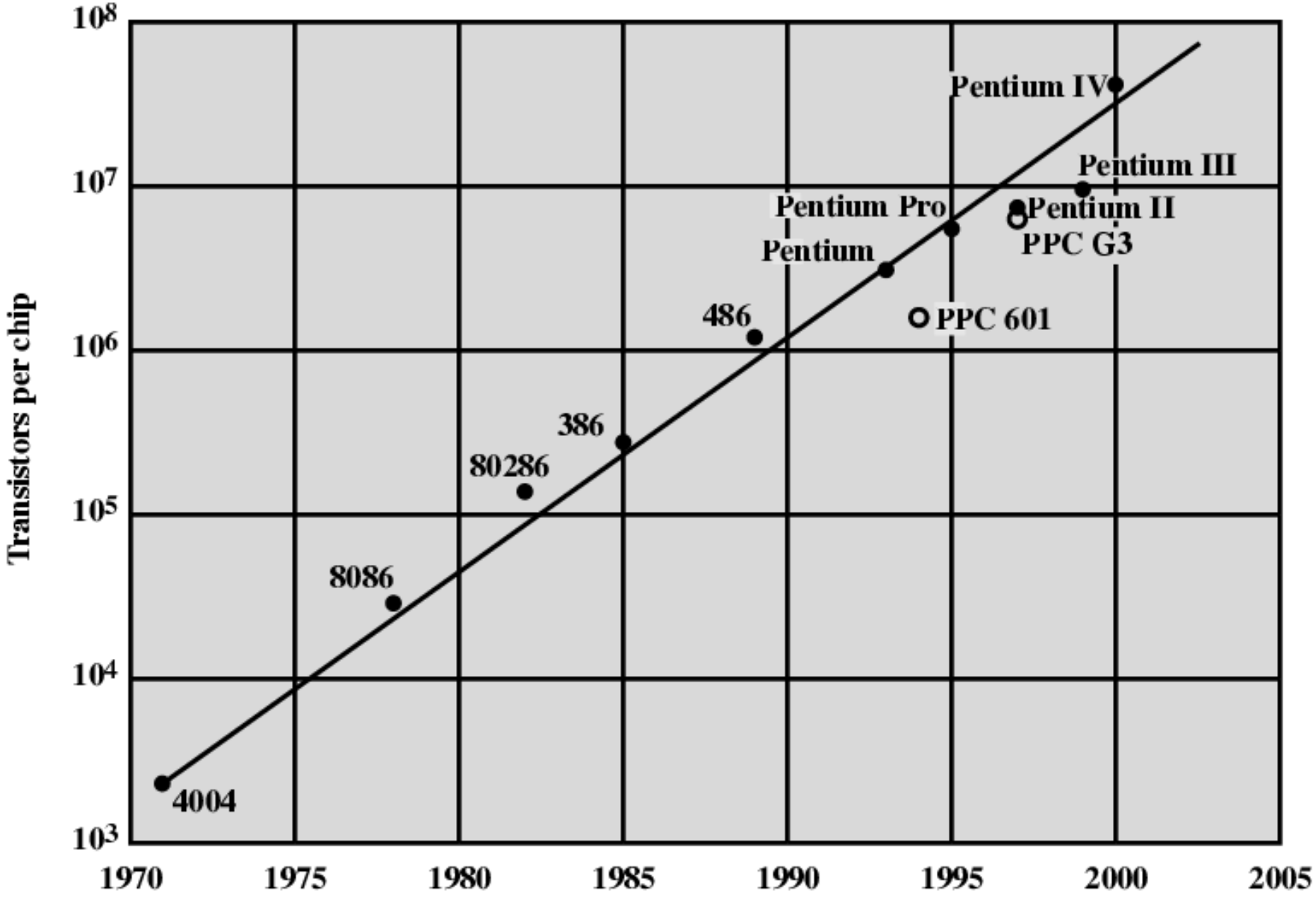
- Literally - "small electronics"
- A computer is made up of gates, memory cells and interconnections
- These can be manufactured on a semiconductor e.g. silicon wafer



Generations of Computer

- Vacuum tube - 1946-1957
- Transistor - 1958-1964
- Small scale integration - 1965 on
  - —Up to 100 devices on a chip
- Medium scale integration - to 1971
  - —100-3,000 devices on a chip
- Large scale integration - 1971-1977
  - —3,000 - 100,000 devices on a chip
- Very large scale integration - 1978 to date
  - —100,000 - 100,000,000 devices on a chip
- Ultra large scale integration
  - —Over 100,000,000 devices on a chip

Growth in CPU Transistor Count



IBM 360 series (1964)

Replaced (&not compatible with) 7000series;1st planned family of computers

- Similar or identical instruction sets
- Similar or identical O/S
- Increasing speed
- Increasing number of I/O ports (i.e. more terminals)
- Increased memory size;
- Increased cost

Characteristic	Model 30	Model 40	Model 50	Model 65	Model 75
Maximum memory size (bytes)	64K	256K	256K	512K	512K
Data rate from memory (Mbytes/sec)	0.5	0.8	2.0	8.0	16.0
Processor cycle time μs)	1.0	0.625	0.5	0.25	0.2
Relative speed	1	3.5	10	21	50
Maximum number of data channels	3	3	4	6	6
Maximum data rate on one channel (Kbytes/s)	250	400	800	1250	1250

System 360

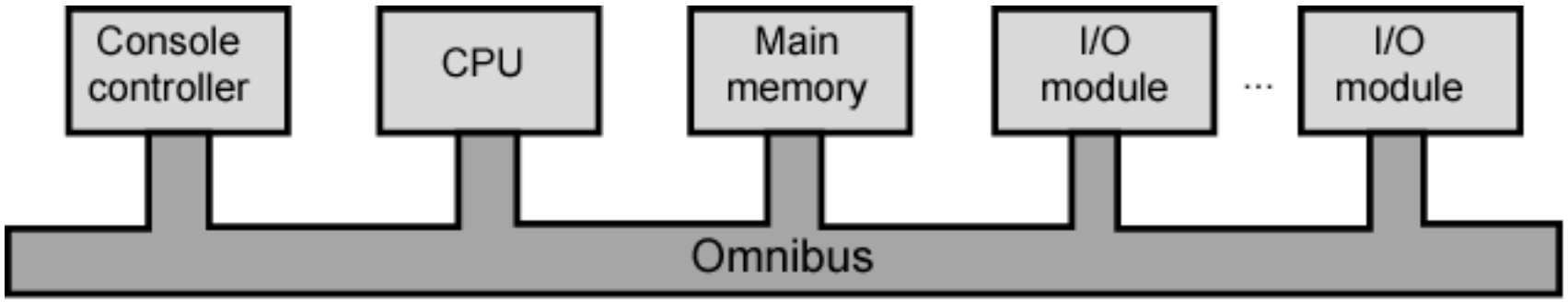
- Differences were:
  - —Complexity of ALU circuitry allowing parallel execution
  - —Increased bus width allowing faster data transfer
- The 360 family gave IBM a huge market share
- Architecture still used for IBM mainframes

DEC PDP-8 (1964)

- First minicomputer
- Did not need air conditioned room
- Small enough to sit on a lab bench
- $16,000
  - —vs. $100k+ for IBM 360
- Embedded applications & OEM
- Introduced BUS STRUCTURE (vs. central-switched IBM architecture)

DEC - PDP-8 Bus Structure

- 96 separate signal paths carry address, data and control signals
- Easy to plug new modules into bus
- Bus is controlled by CPU
- Now virtually universal in microcomputers

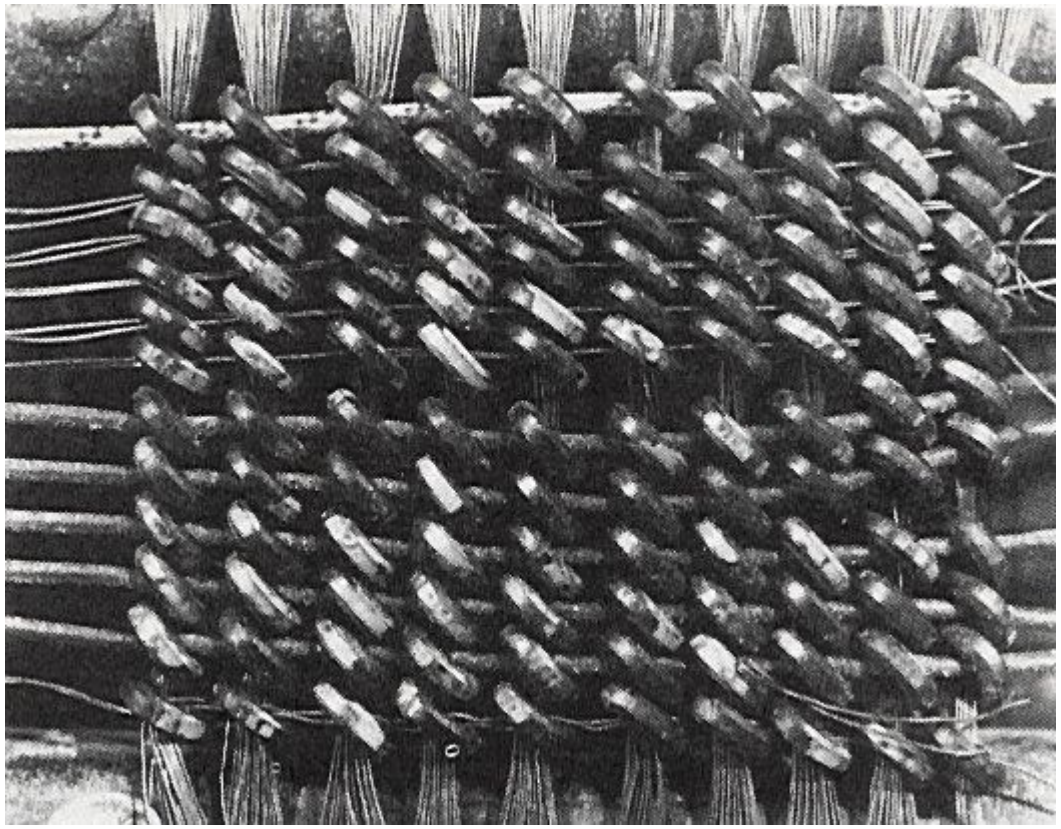


Beyond 3rd Generation

- Classification by generation became less meaningful after the early 1970's due to rapid pace of technological development
- Two main developments are still being worked out:
  - Semiconductor memory
  - Microprocessors

Magnetic Core Memory

- Mostly assembled by hand;
- Each bit stored in a small iron ring(core, about 1/16” diameter); magnetized one way:1, other way:0
- Reads are destructive; restore needed
- See http://www.columbia.edu/acis/history/core.html



Semiconductor Memory

- First developed in 1970 by Fairchild
- Chip size = size of a single core (i.e. 1 bit of magnetic core storage)
- Holds 256 bits
- Non-destructive read
- Much faster than core
- 1974, cheaper than core memory (price per bit)
- Capacity (density of elements) approximately doubles each year
- Gens:1K,4K,16K,64K,256K,1M, 4M, 16M, 64M, 256M
- GBs chips now available

Intel Microprocessors

- 1971 - 4004
  - —First microprocessor
  - —All CPU components on a single chip
  - —4 bit; add, multiply=repeated add
- Followed in 1972 by 8008
  - —8 bit
  - —Both designed for specific applications
- 1974 - 8080
  - —Intel's first general purpose microprocessor
  - —8-bit
  - —Richer instruction set
  - —Large addressing capability

16 and 32 bit processors

- 16-bit processors (8086) were introduced in the late 1970's
- First 32-processors were introduced in 1981 (Bell Labs and Hewlett-Packard)
- Intel's 32-bit processor did not arrive until 1985 (80386)

- 64-bit processor: Pentium Pro (1995), ...

Intel x86 Processors 1971 - 1989

(a) 1970s Processors

	4004	8008	8080	8086	8088
Introduced	1971	1972	1974	1978	1979
Clock speeds	108 kHz	108 kHz	2 MHz	5 MHz, 8 MHz, 10 MHz	5 MHz, 8 MHz
Bus width	4 bits	8 bits	8 bits	16 bits	8 bits
Number of transistors	2,300	3,500	6,000	29,000	29,000
Feature size (μm)	10		6	3	6
Addressable memory	640 Bytes	16 KBytes	64 KBytes	1 MB	1 MB
Virtual memory	—	—	—	—	—

(b) 1980s Processors

	80286	386TM DX	386TM SX	486TM DX CPU
Introduced	1982	1985	1988	1989
Clock speeds	6 MHz - 12.5 MHz	16 MHz - 33 MHz	16 MHz - 33 MHz	25 MHz - 50 MHz
Bus width	16 bits	32 bits	16 bits	32 bits
Number of transistors	134,000	275,000	275,000	1.2 million
Feature size (μm)	1.5	1	1	0.8 - 1
Addressable memory	16 megabytes	4 gigabytes	16 megabytes	4 gigabytes
Virtual memory	1 gigabyte	64 terabytes	64 terabytes	64 terabytes

Intel x86 Processors 1990 - 2002

(c) 1990s Processors

	486TM SX	Pentium	Pentium Pro	Pentium II
Introduced	1991	1993	1995	1997
Clock speeds	16 MHz - 33 MHz	60 MHz - 166 MHz,	150 MHz - 200 MHz	200 MHz - 300 MHz
Bus width	32 bits	32 bits	64 bits	64 bits
Number of transistors	1.185 million	3.1 million	5.5 million	7.5 million
Feature size (μm)	1	0.8	0.6	0.35
Addressable memory	4 gigabytes	4 gigabytes	64 gigabytes	64 gigabytes
Virtual memory	64 terabytes	64 terabytes	64 terabytes	64 terabytes

(d) Recent Processors

	Pentium III	Pentium 4	Itanium	Itanium 2
Introduced	1999	2000	2001	2002
Clock Speeds	450 - 660 MHz	1.3 - 1.8 GHz	733 - 800 MHz	900 MHz - 1 GHz
Bus Width	64 bits	64 bits	64 bits	64 bits
Number of Transistors	9.5 million	42 million	25 million	220 million
Feature size (μm)	0.25	0.18	0.18	0.18
Addressable Memory	64 gigaBytes	64 gigaBytes	64 gigaBytes	64 gigaBytes
Virtual Memory	64 teraBytes	64 teraBytes	64 teraBytes	64 teraBytes

Processing Power

- Today's $400 personal computer has more computing power than a 1990 mainframe
- Much of this is gobbled up by bloated operating systems (e.g., Windows)
- But many PC applications are only possible with great processing capability:
  - —Image processing
  - —Video Editing
  - —Speech recognition
  - —Simulation modeling
  - —Multimedia authoring
  - —Servers for database and transaction processing

Design for Performance: Speeding it up

Some techniques for performance increase

- Pipelining
- On board cache
- On board L1 & L2 cache
- Branch prediction
- Data flow analysis
- Speculative execution

Pipelining: Idea

- Increases the Throughput of the Processor
- Don’t focus on trying to speedup individual instructions
- Instead focus on throughput i.e. number of instructions executed per unit time
- Like an assembly line
- Instruction execution is divided into stages
- When one stage has completed, that circuitry is free to operate on the “next” instruction
- At any moment, multiple instructions get processed, each instruction on a different stage.

Pipelining Example

- Consider a sandwich shop with a five step process
  - Take order
  - Bread
  - Cheese
  - Meat
  - Veggies
- One employee can do the job
- Now imagine 5 employees making sandwiches

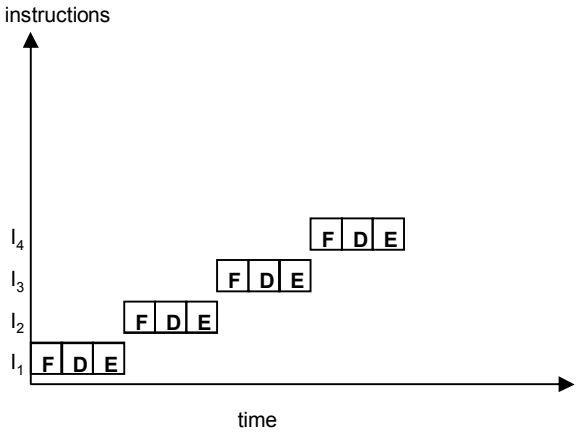
Pipeline Math

- If it takes one person 5 minutes to make a sandwich
- And we pipeline the process using 5 people each taking a minute
- And we start making sandwiches constantly (i.e. ignore startup pipeline filling)
- How long does it actually take to make a single sandwich (Real elapsed time)
- What is the effective time to produce a sandwich? (i.e. a sandwich exits from the pipeline every how many minutes?)

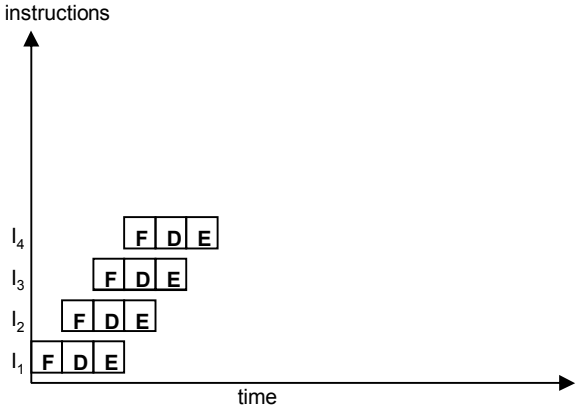
Towards an instruction processing assembly line

Macro State	Functional Units in Use			
FETCH	IR	ALU	PC	MEM
DECODE		IR		
EXECUTE (ADD)		IR	ALU	Reg-file

Without Pipeline



With Pipeline



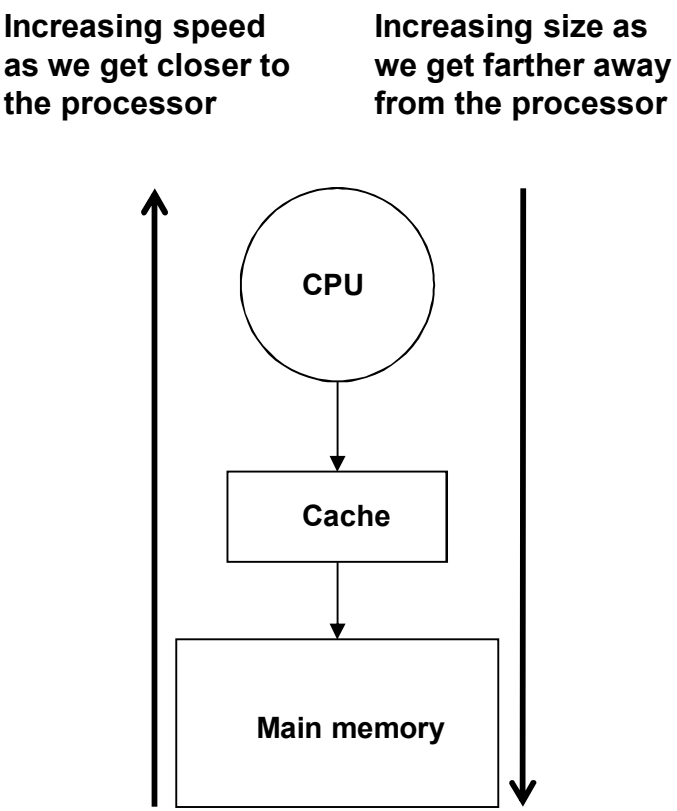
Cache Memory

- Reality…
- Processors have cycle times of ~1 ns
- Fast DRAM has a cycle time of ~100 ns
- We have to bridge this gap to be effective!
- Idea: Very fast memory stores copies of data in slow main memory
- Can be onboard (same chip as processor) or in separate chips

The Concept of a Cache

- Feasible to have small amount of fast memory and/or large amount of slow memory.

- CPU looks in cache for data it seeks from main memory.
- If data not there it retrieves it from main memory.
- If the cache is able to service "most" CPU requests then effectively we will get speed advantage of cache.
- All addresses in cache are also in memory



Branch Prediction

- Processor looks ahead in instruction stream and attempts to predict which branch will be taken by conditional code
- Execute predicted branch and hope it's right
- Wrong guess leads to pipeline stall

Data Flow Analysis

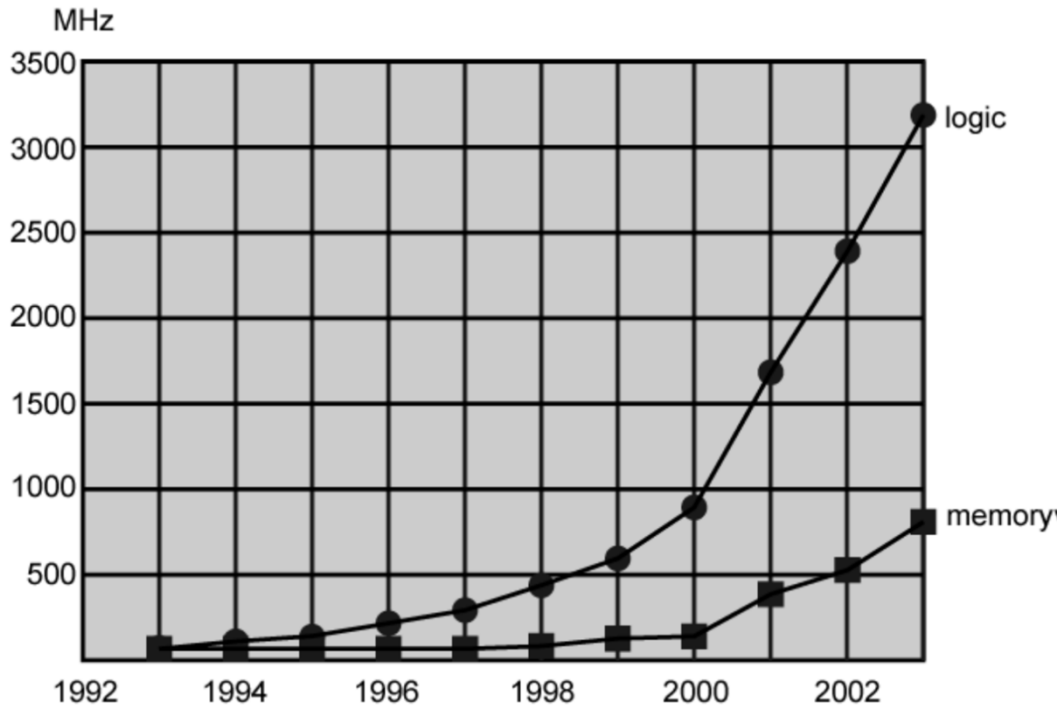
- Processor examines code to determine which instructions are dependent on results of instructions in pipeline
- Reorders instructions in code so that independent instructions can be executed in parallel pipelines

Speculative Execution

- With branch prediction and data flow analysis processor can execute some instructions before they appear in the execution stream
- Results stored in temporary locations

Performance Mismatch

- Processor speed increased
- Memory capacity increased
- BUT: Memory speed lags behind processor speed



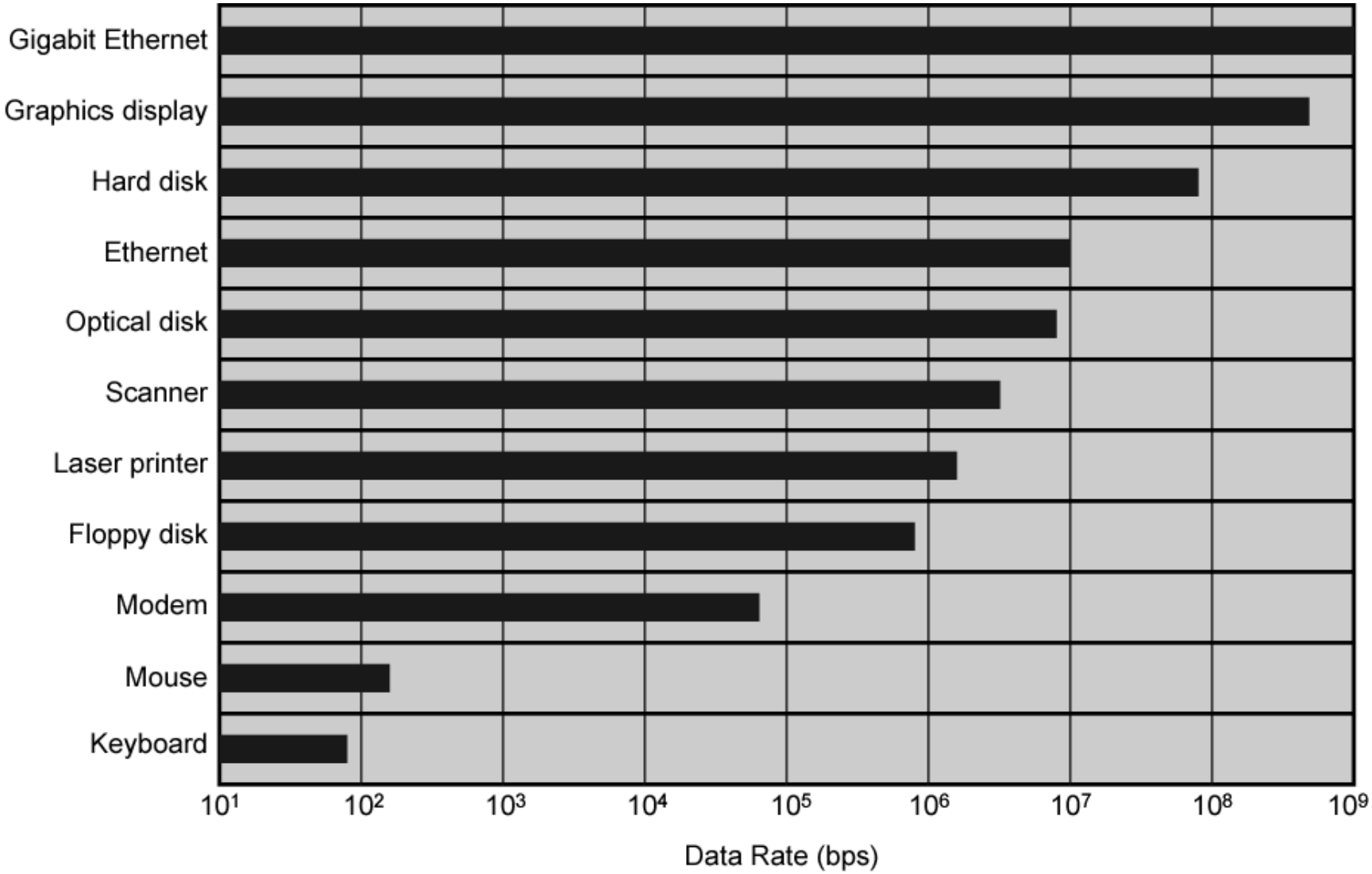
Solution

- Increase number of bits retrieved at one time
  - —Make DRAM “wider” rather than “deeper”
  - —**DDR(double data rate), DDR2, DDR3, DDR4**
- Change DRAM interface
  - —Cache
- Reduce frequency of memory access
  - —More complex cache and cache on chip
- Increase interconnection bandwidth
  - —High speed buses
  - —Hierarchy of buses

I/O Devices

- Peripherals with intensive I/O demands
- Large data throughput demands
- Processors can handle this
- But there are problems moving data between processor and peripheral
- Solutions:
  - —Caching
  - — Buffering
  - —Higher-speed interconnection buses
  - —More elaborate bus structures
  - —Multiple-processor configurations

Typical I/O Device Data Rates



Key is Balance

- Processor components
- Main memory
- I/O devices
- Interconnection structures

Increased Cache Capacity

- Typically two or three levels of cache between processor and main memory
- Chip density increased
  - More cache memory on chip
  - Faster cache access
- Pentium chip devoted about 10% of chip area to cache
- Pentium 4 devotes about 50%

More Complex Execution Logic

- Enable parallel execution of instructions
- Pipeline works like assembly line
  - Different stages of execution of different instructions at same time along pipeline
- Superscalar processing allows multiple pipelines within single processor
  - Instructions that do not depend on one another can be executed in parallel

Diminishing Returns

- Internal organization of processors complex
  - —Can get a great deal of parallelism
  - —Further significant increases likely to be relatively modest
- Benefits from cache are reaching limit
- Increasing clock rate runs into power dissipation problem
  - —Some fundamental physical limits are being reached

New Approach – Multiple Cores

- Multiple processors on single chip
  - —Large shared cache
- Within a processor, increase in performance proportional to square root of increase in complexity
- If software can use multiple processors, doubling number of processors almost doubles performance
- So, use two simpler processors on the chip rather than one more complex processor
- With two processors, larger caches are justified
  - —Power consumption of memory logic less than processing logic
- Example: IBM POWER4
  - — Two cores based on PowerPC