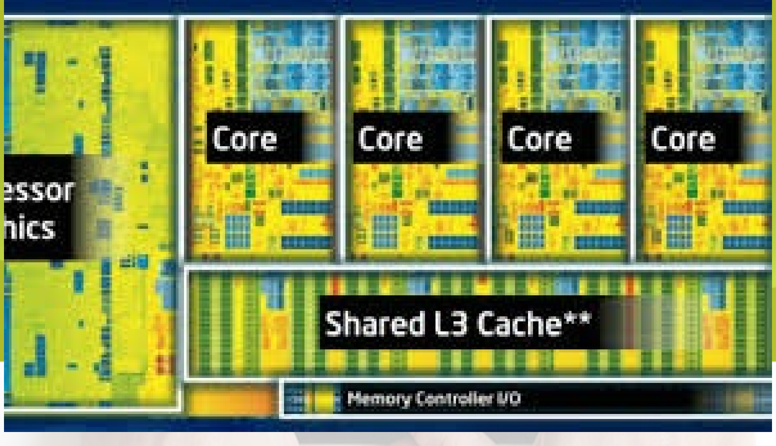
The Central Processing Unit:

What Goes on Inside the Computer

Lecture 2 Chapter 1

CPU



Microprocessor

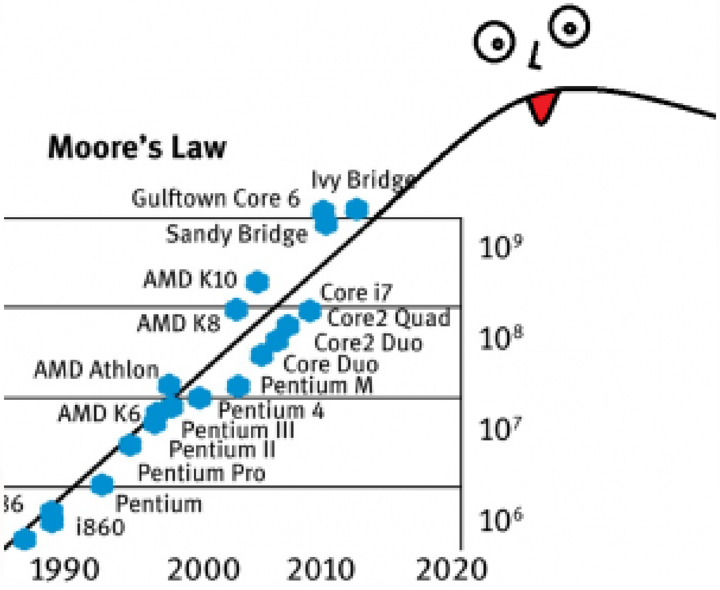
- CPU etched on a chip
- Chip size is ¼ x ¼ inch
- Composed of silicon
- Contains millions of transistors
  - Electronic switches that can allow current to pass through

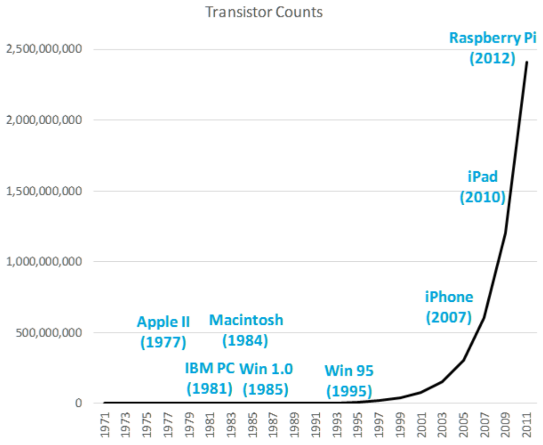
Moore Low

- Moore’s law states that the number of transistors on a chip doubles every 18 months.
- This ‘‘law’’ is an observation by Intel cofounder Gordon Moore of how fast process engineers at the semiconductor companies are able to shrink their transistors.

- Moore’s law has held for over three decades now and is expected to hold for at least one more.

- After that, the number of atoms per transistor will become too small and quantum mechanics will start to play a big role, preventing further shrinkage of transistor sizes.





Building a Better Microprocessor

- Computers imprint circuitry onto microchips
  - Cheaper
  - Faster
- Perform functions of other hardware
  - Math coprocessor is now part of microprocessor
  - Multimedia instructions are now part of microprocessor

Building a Better Microprocessor

The more functions that are combined on a microprocessor:

- The faster the computer runs
- The cheaper it is to make
- The more reliable it is

Types of Microprocessors

Intel

- Pentium
- Celeron
- Xeon and Itanium

Intel-compatible

- Cyrix
- AMD

Types of Microprocessors

- PowerPC
  - Cooperative efforts of Apple, IBM, and Motorola
  - Used in Apple Macintosh family of PCs
  - Found in servers and embedded systems
- Alpha
  - Manufactured by Compaq
  - High-end servers and workstations

Microprocessor Components

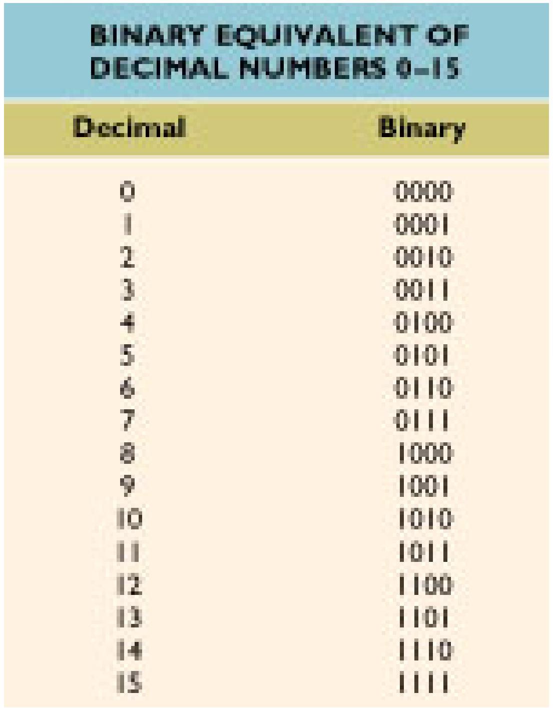
- Control Unit – CU
- Arithmetic / Logic Unit – ALU
- Registers
- System clock

Data Representation *On/Off*

Binary number system is used to represent the state of the circuit

1 0

ON OFF



BINARY EQUIVALENT OF DECIMAL NUMBERS 0–15

Decimal	Binary
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111

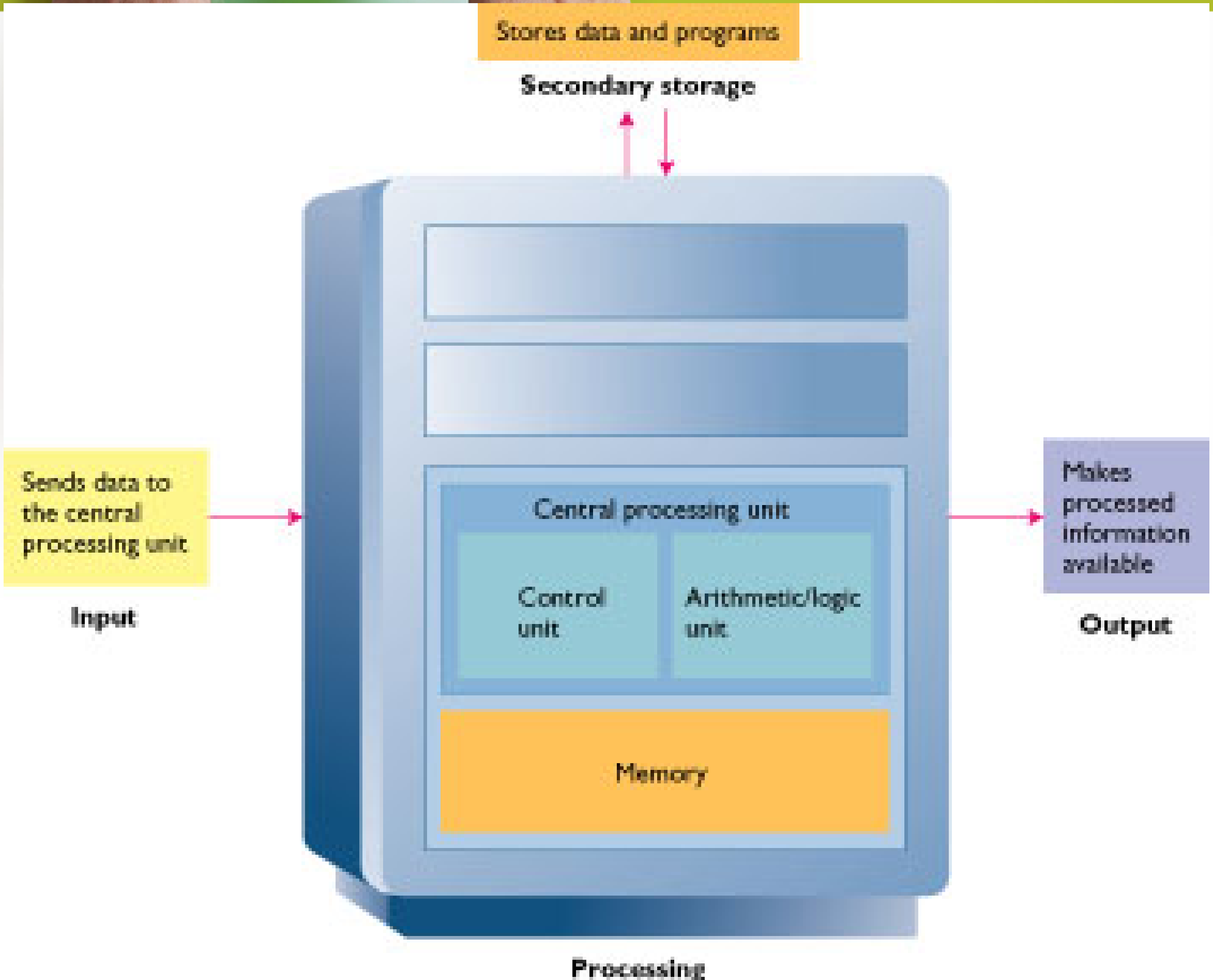
Bits, Bytes, Words

- BIT
  - **B**inary Dig**IT**
  - On/off circuit
  - 1 or 0
- BYTE
  - 8 bits
  - Store one alphanumeric character
- WORD
  - Size of the register
  - Number of BITS that the CPU processes as a unit

Coding Schemes

- ASCII
  - Uses one 8 bit byte
  - 2^8 = 256 possible combinations or characters
  - Virtually all PCs and many larger computers
- EBCDIC
  - Uses one 8 bit byte
  - 28 =256 possible combinations or characters
  - Used primarily on IBM-compatible mainframes
- Unicode
  - Uses two 8 bit bytes (16 bits)
  - 216 = 65,536 possible combinations or characters
  - Supports characters for all the world’ s languages
  - Downward-compatible with ASCII

The CPU


Stores data and programs
Secondary storage
Sends data to the central processing unit
Input
Central processing unit
Control unit
Arithmetic/logic unit
Memory
Makes processed information available
Output
Processing

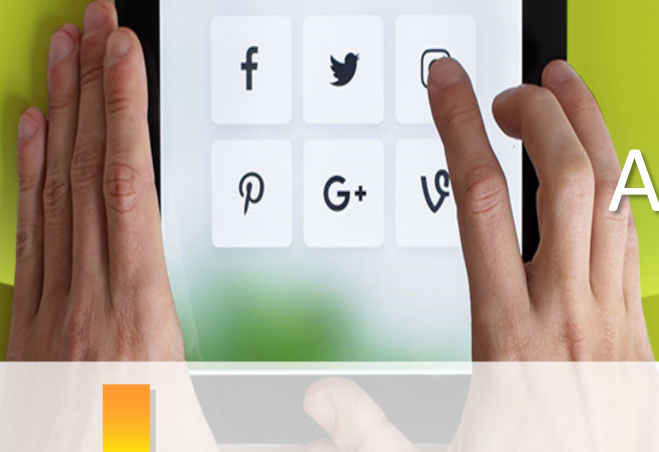
Control Unit
CU

- Part of the hardware that is in-charge
- Directs the computer system to execute stored program instructions
- Communicates with other parts of the hardware

Arithmetic / Logic Unit
ALU

Performs arithmetic operations

Performs logical operations



Arithmetic Operations

+ −

Addition

Subtraction

Multiplication

Division

/ *

Logical Operations

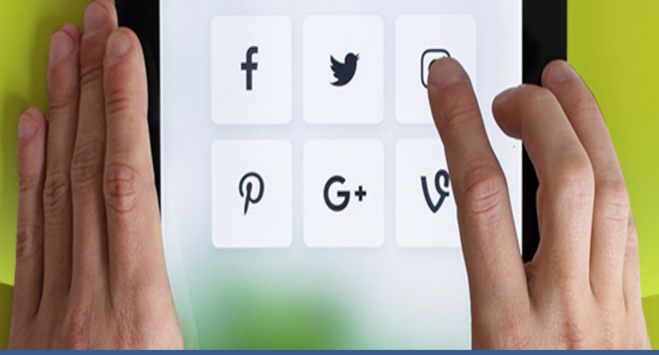
- Evaluates conditions
- Makes comparisons
- Can compare
  - Numbers
  - Letters
  - Special characters

NOT

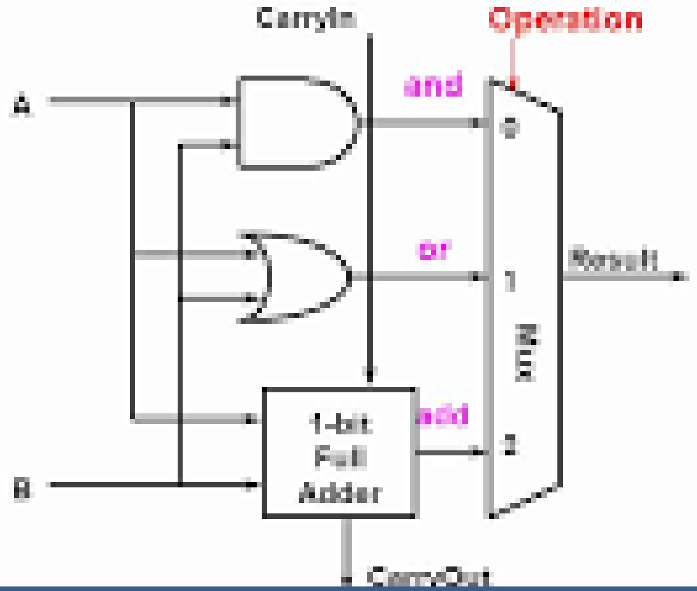
AND

OR

= > < <> >= <=



1 Bit ALU


CarryIn
Operation
A
and
0
or
1
Result
Mux
add
2
1-bit
Full
Adder
B
CarryOut

Registers

Special-purpose

High-speed

Temporary storage

Located inside CPU

Instruction register

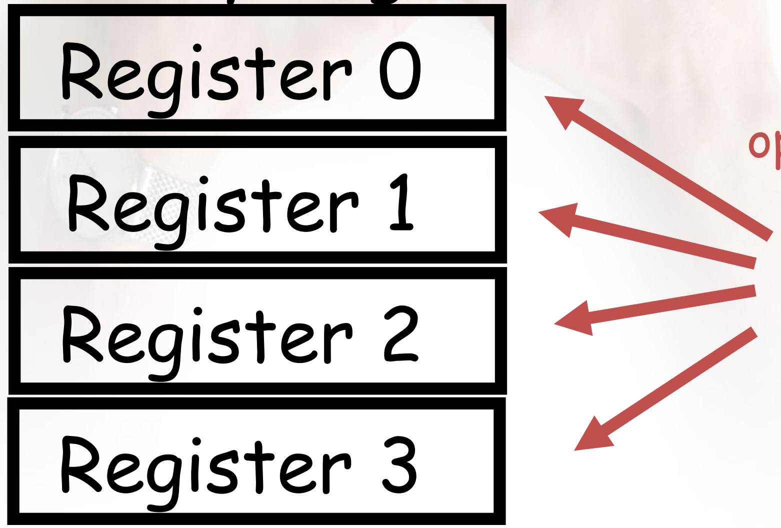
Holds instruction currently being executed

Data register

Holds data waiting to be processed

Holds results from processing

Inside the CPU



Memory Registers

Register 0

Register 1

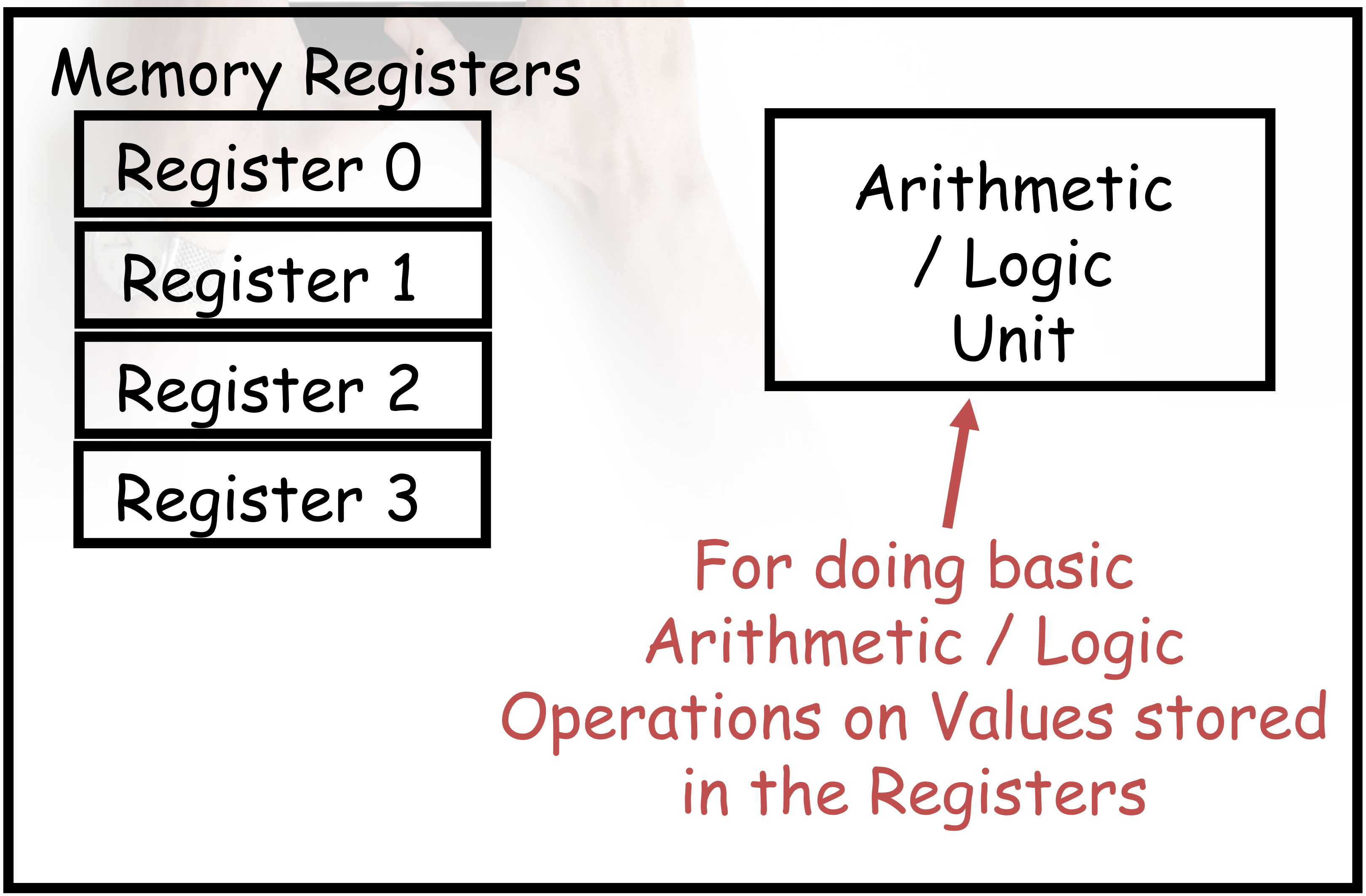
Register 2

Register 3

Temporary Memory. Computer "Loads" data from RAM to registers, performs operations on data in registers, and "stores" results from registers back to RAM

Remember our initial example: "read value of A from memory; read value of B from memory; add values of A and B; put result in memory in variable C." The reads are done to registers, the addition is done in registers, and the result is written to memory from a register.

Inside the CPU (cont.)



Memory Registers

Register 0

Register 1

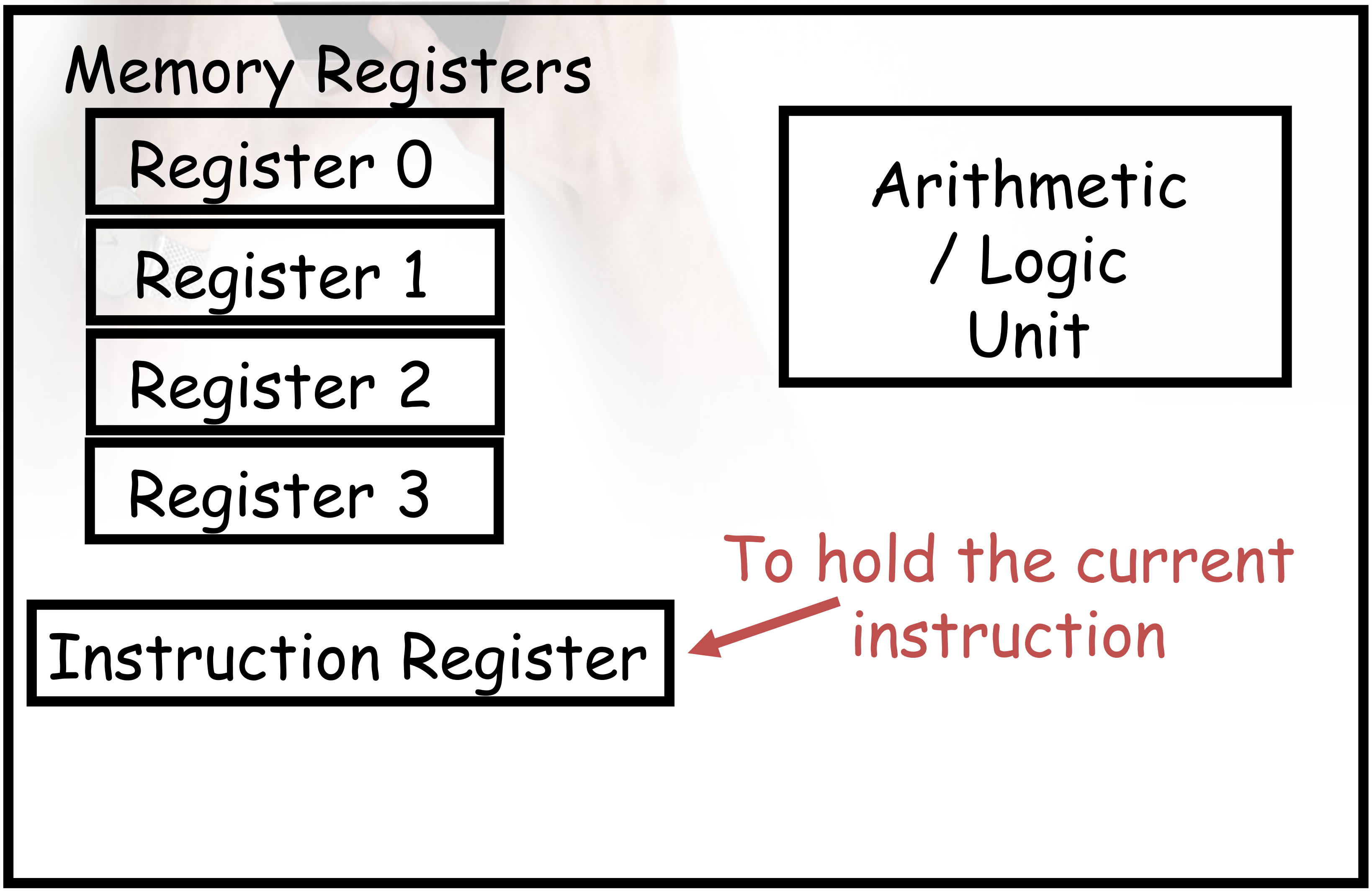
Register 2

Register 3

Arithmetic / Logic Unit

For doing basic Arithmetic / Logic Operations on Values stored in the Registers

Inside the CPU (cont.)



Memory Registers

Register 0

Register 1

Register 2

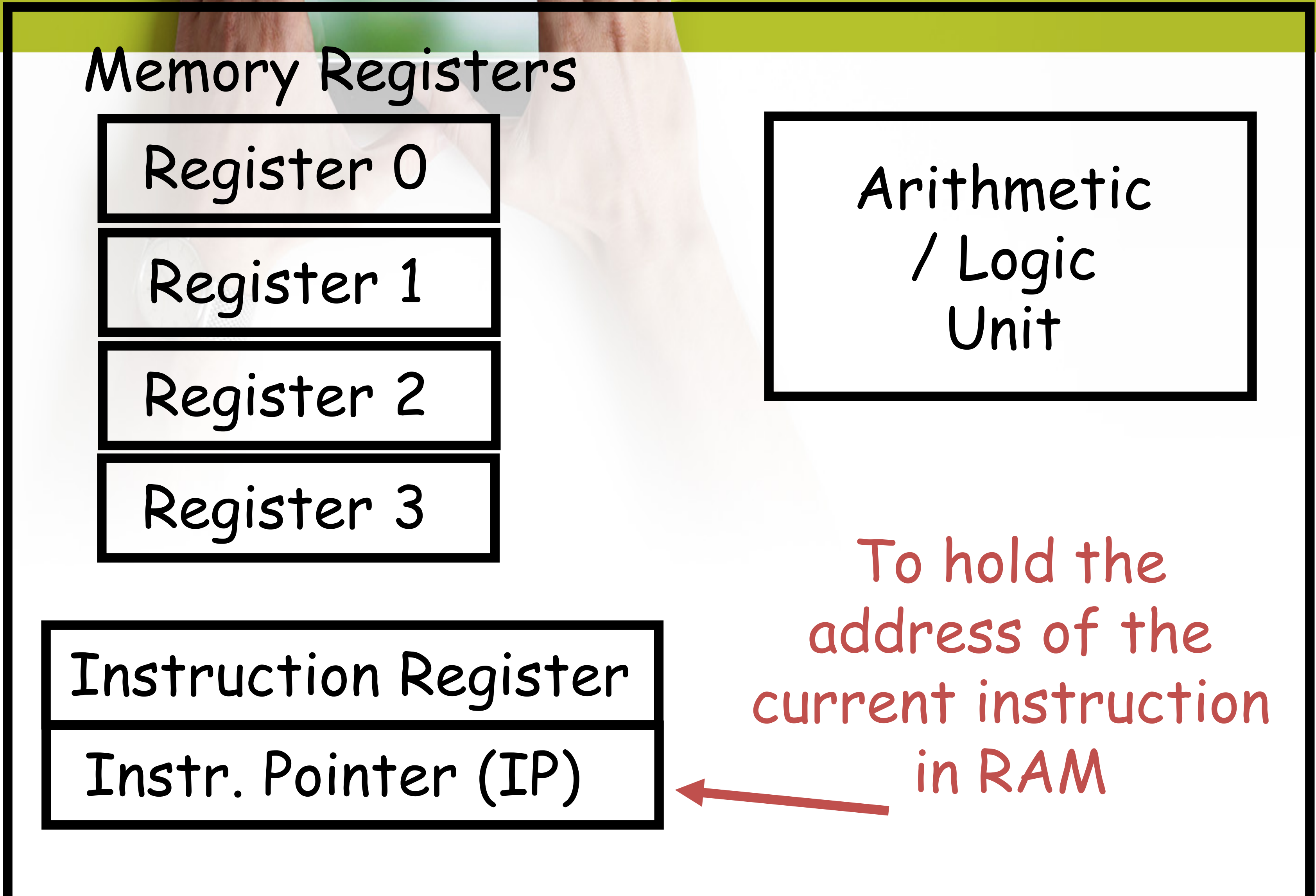
Register 3

Arithmetic / Logic Unit

Instruction Register

To hold the current instruction

Inside the CPU (cont.)



Memory Registers

Register 0

Register 1

Register 2

Register 3

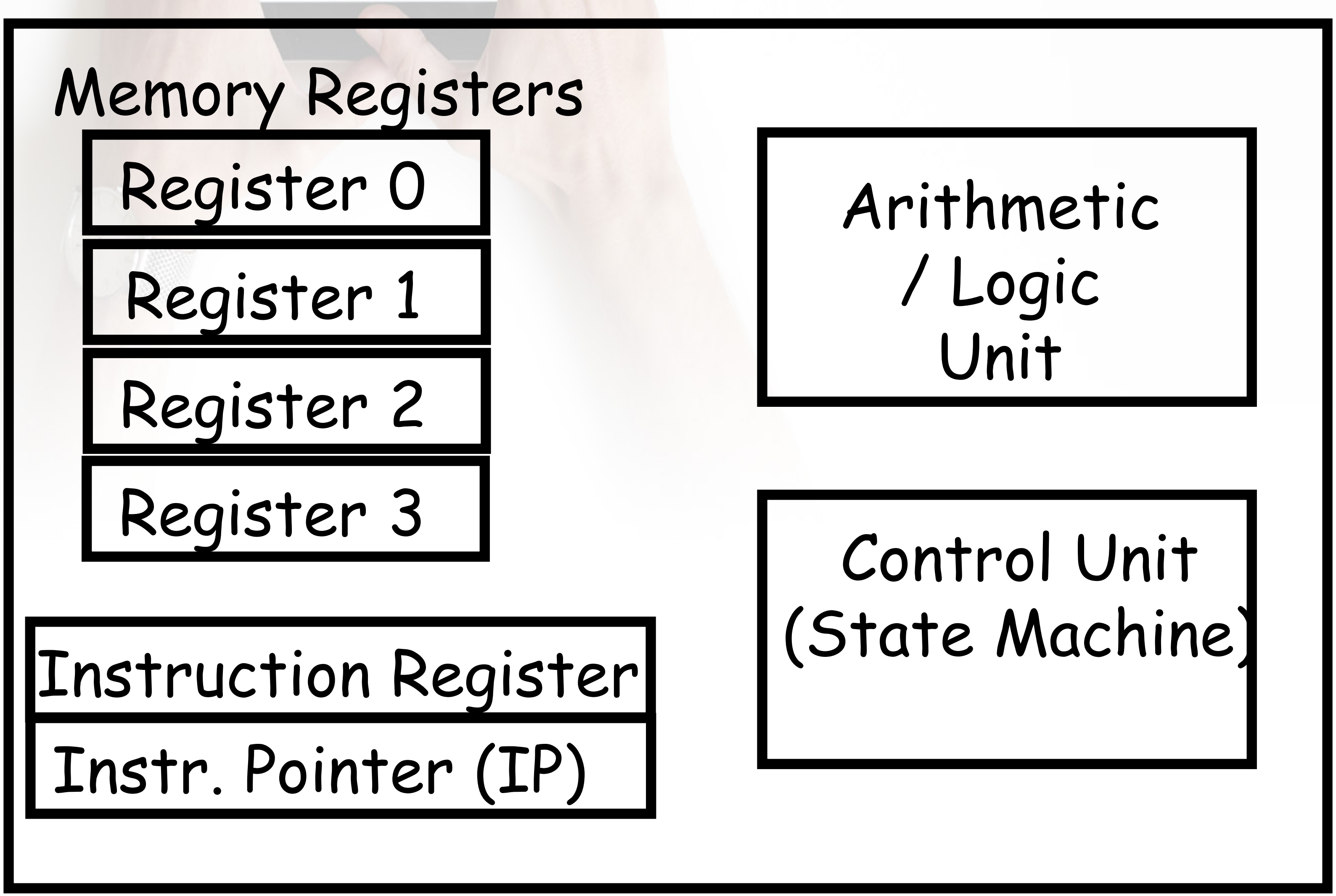
Arithmetic / Logic Unit

Instruction Register

Instr. Pointer (IP)

To hold the address of the current instruction in RAM

Inside the CPU (cont.)



Memory Registers

Register 0

Register 1

Register 2

Register 3

Instruction Register

Instr. Pointer (IP)

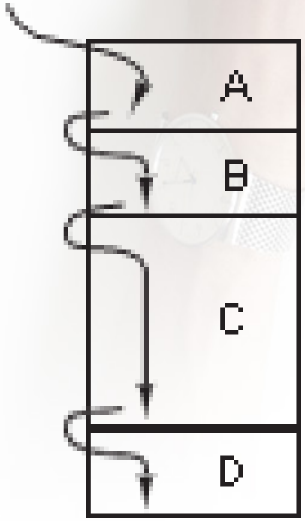
Arithmetic / Logic Unit

Control Unit (State Machine)

The Process Model

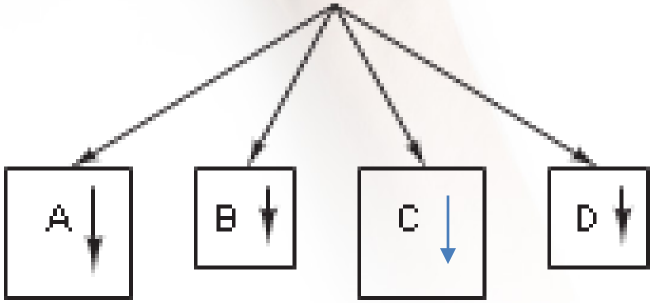
- A process is just an instance of an executing program, including the current values of program counter, registers, and variables.
- The CPU switches from program to program.
- This rapid switching back and forth is called multiprogramming.

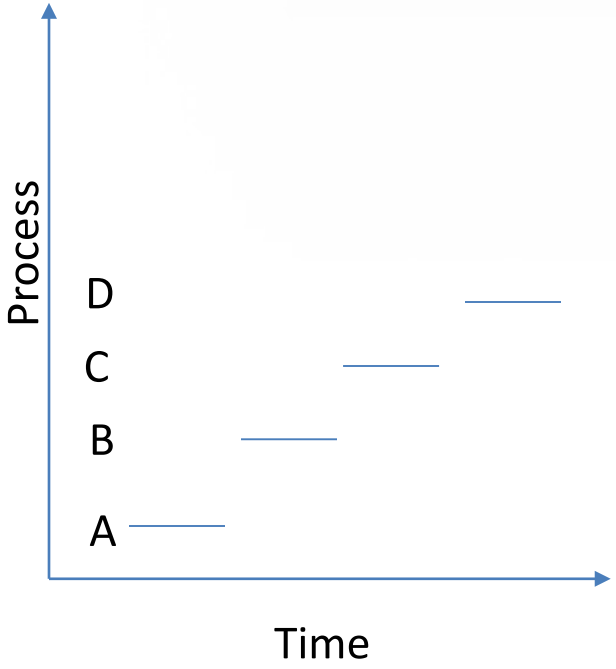
Multiprogramming four Programs



One Program Counter

Four Program Counter





A CPU can really run only ne process at a time, if there are 2 cores Each of them can run only one process at a time

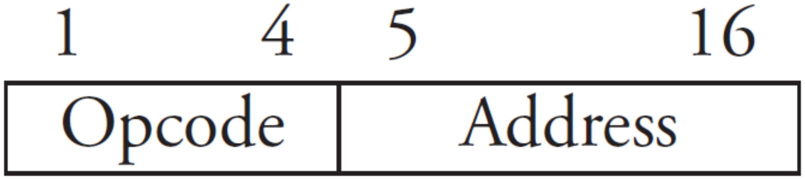
The Register Set

A CPU is a sequential circuit

- repeatedly reads and executes an instruction from its memory
- fetch-and-execute cycle

A ***machine language*** program is a set of instructions drawn from the CPU instruction set

- Each instruction consists of two parts: an **opcode** and an **address**



1 4	5 16
Opcode	Address

Exemple 16 bits

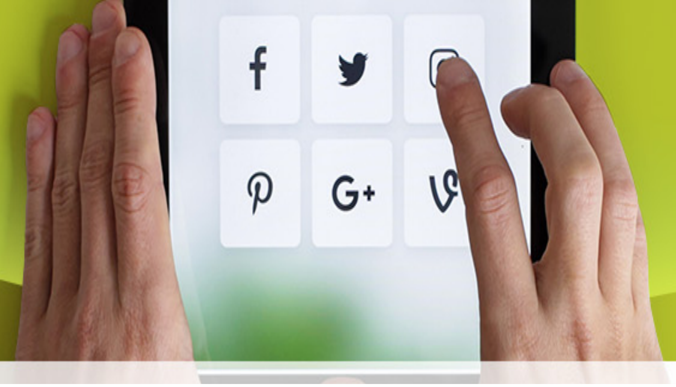
11 instructions require a 4-bit opcode.

12 bits of the 16-bit word remain for addressing $2^{12} = 4096$

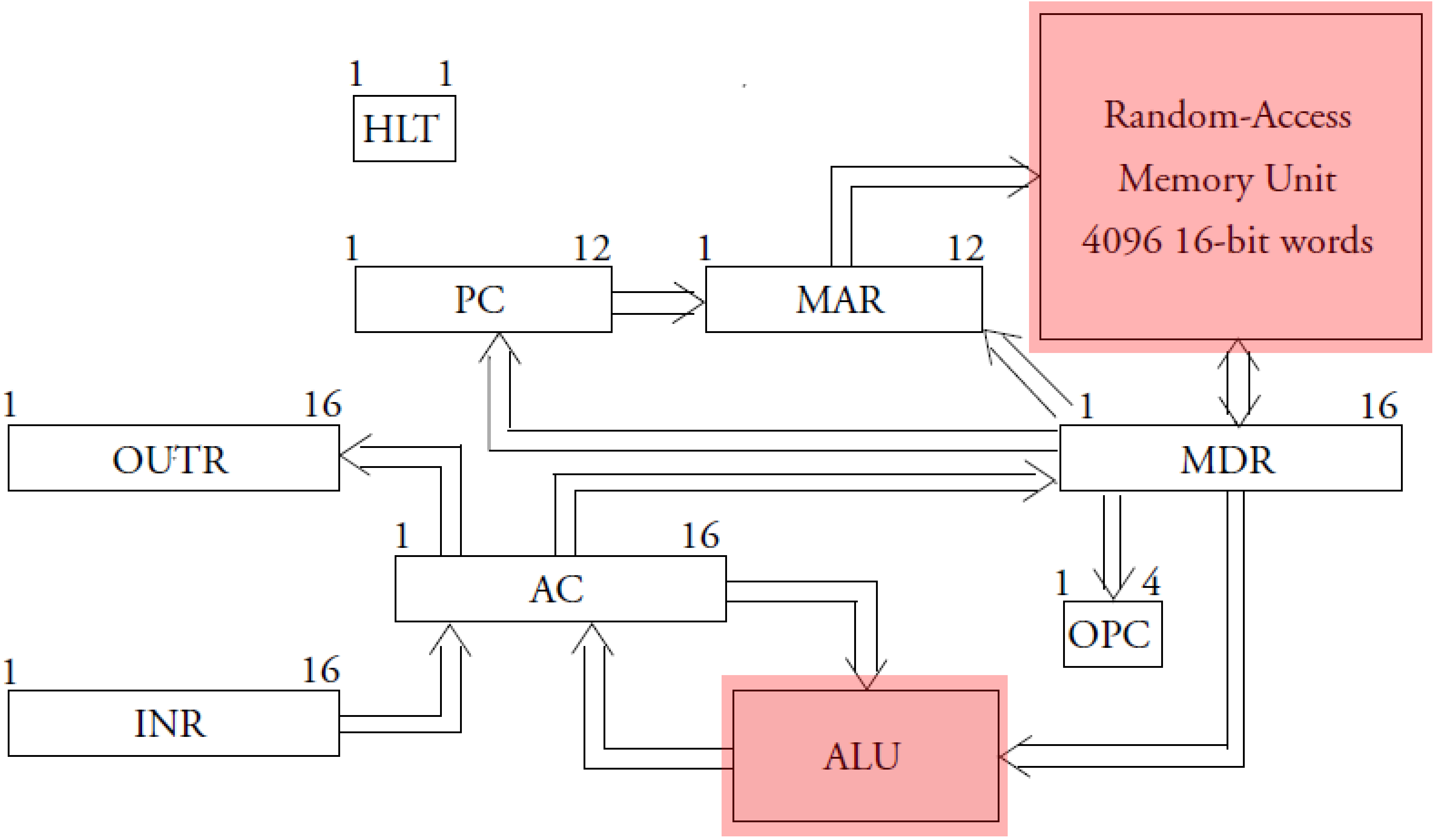
16-bit words of the RAM.

The CPU has 8 special registers:

- 16-bit ***accumulator*** (AC)
- 12-bit ***program counter*** (PC)
- 4-bit ***opcode*** (OPC)
- 12-bit ***memory address register*** (MAR)
- 16-bit ***memory data register*** (MDR)
- 16-bit ***input register*** (INR)
- 16-bit ***output register*** (OUTR)
- 1-bit ***halt register*** (HLT)



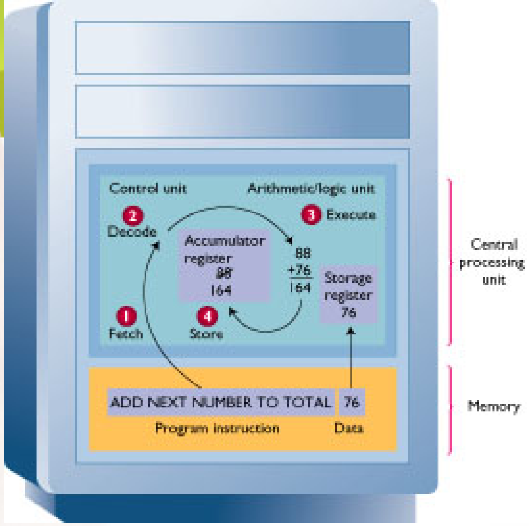
The Registers Data Transfer in the CPU



The Control Unit

- It all comes down to the Control Unit.
- This is just a State Machine.
- How does it work?

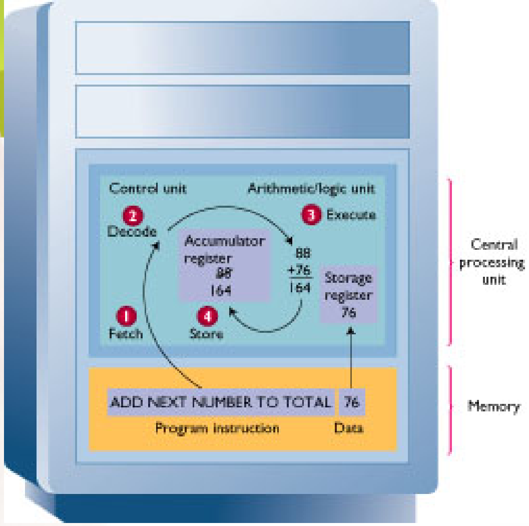
Machine Cycle



I-time

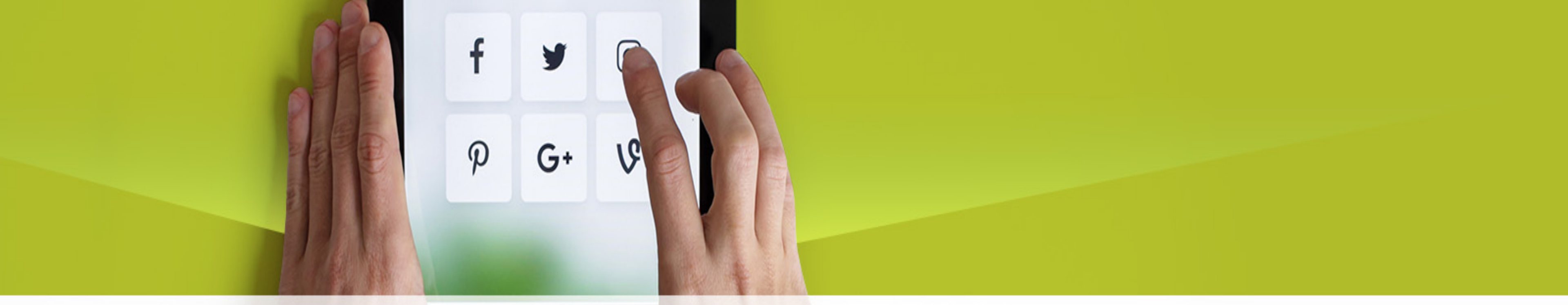
- CU fetches an instruction from memory and puts it into a register
- CU decodes the instruction and determines the memory location of the data required

Machine Cycle



E-time

- Execution
  - CU moves the data from memory to registers in the ALU
  - ALU is given control and executes the instruction
  - Control returns to the CU
- CU stores the result of the operation in memory or in a register



Direct memory instruction

- Addresses refer directly to memory locations containing the ***operands*** (data) on which the program operates

Indirect memory instructions

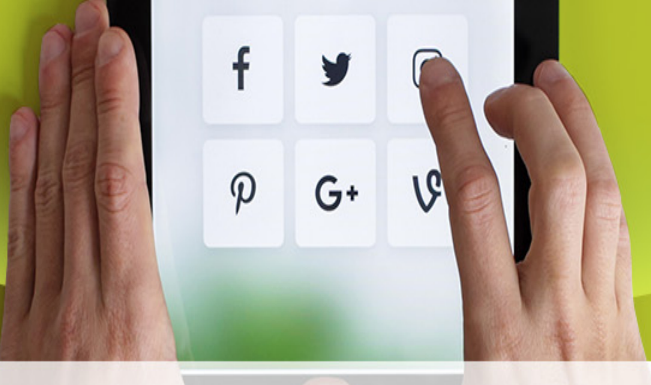
- Instructions in which an address is interpreted as the address at which to find the address containing the needed operand
- CPU does two memory fetches
  - the first to find the address of the operand
  - the second to find the operand itself

The Fetch-and-Execute Cycle

- Fetch-and-execute cycle has two portions
- instruction, whose address is in the PC, is fetched into the MDR
- opcode portion of this register is copied into the OPC

The Control Unit

- Control Unit State Machine has very simple structure:
  - 1) Fetch: Ask the RAM for the instruction whose address is stored in IP.
  - 2) Execute: There are only a small number of possible instructions. Depending on which it is, do what is necessary to execute it.
  - 3) Repeat: Add 1 to the address stored in IP, and go back to Step 1 !



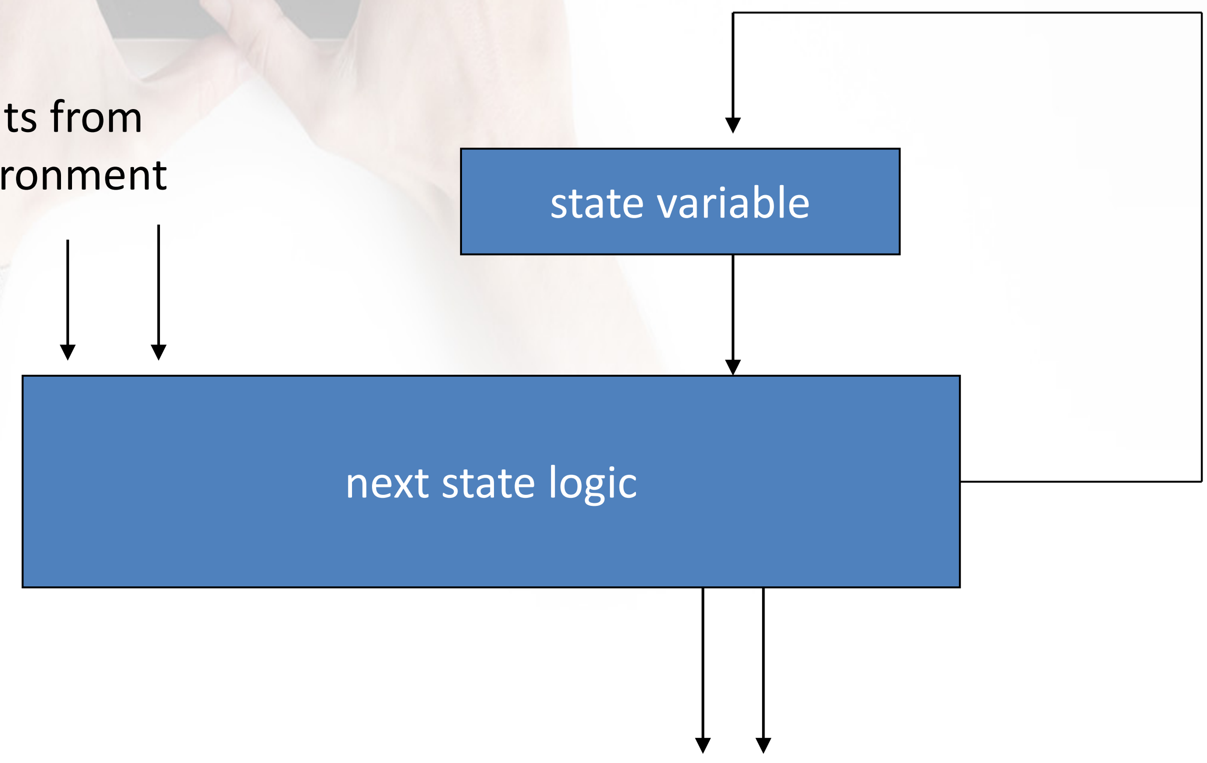
What is a FSM

- A Finite State Machine (FSM) is based on the idea of there being finite number of states for a given system.
- For instance, when an application turns an LED on and off, two states exist; one state is when the LED is on and the other is when it is off.
- If it will turn on eight LEDs sequentially.
- Only one LED is on at a time, therefore eight states exist.
- Each state consists of one LED being turned on while all the rest are off.

State variable

- State machines require a State Variable (SV).
- The SV is essentially a pointer that keeps track of the state that the system is in.
- The SV can be modified in the software modules (or states) themselves or by an outside function.
- The example firmware uses an outside function which detects a button press to advance through the states.

State machine

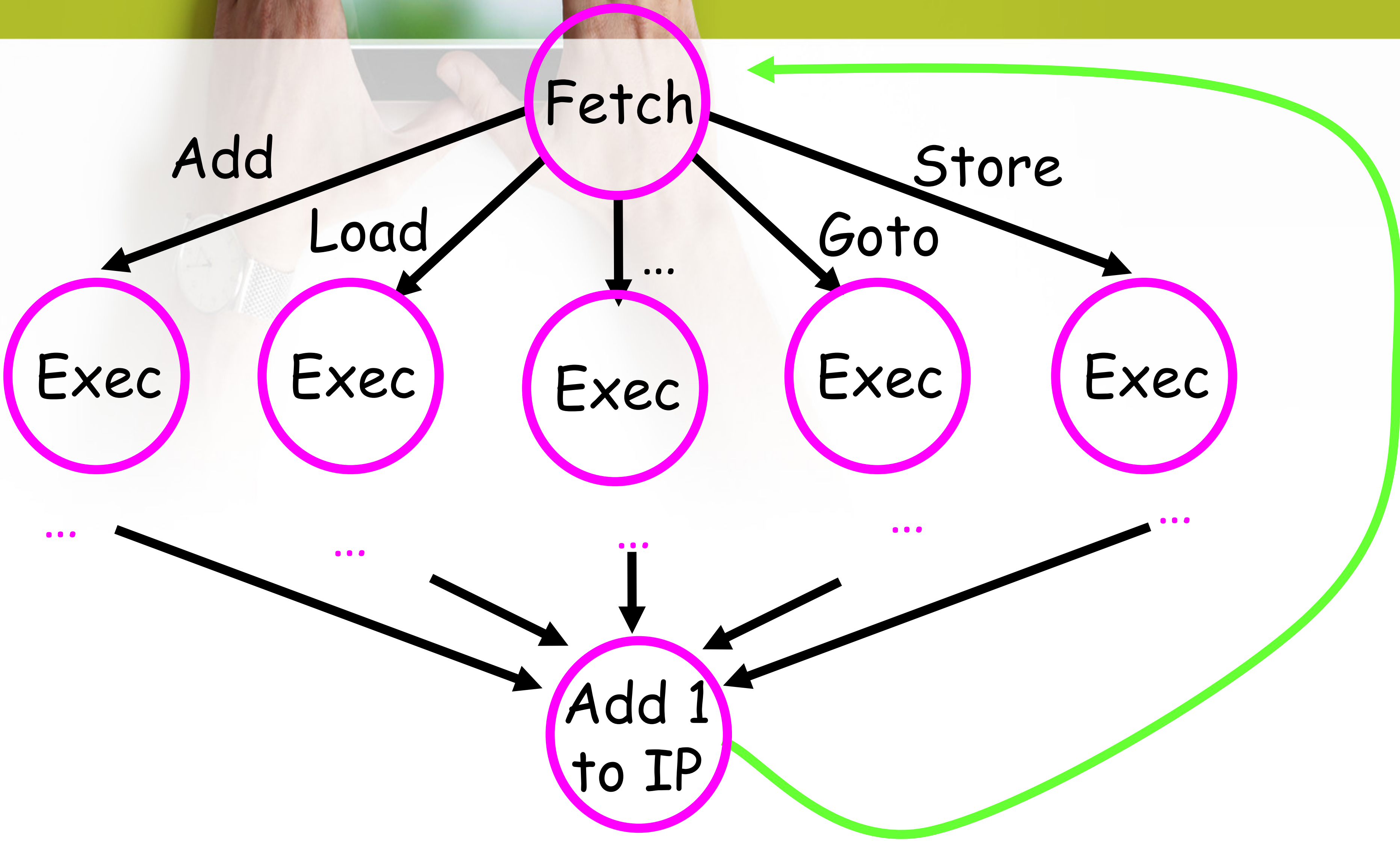


Next state

inputs from environment

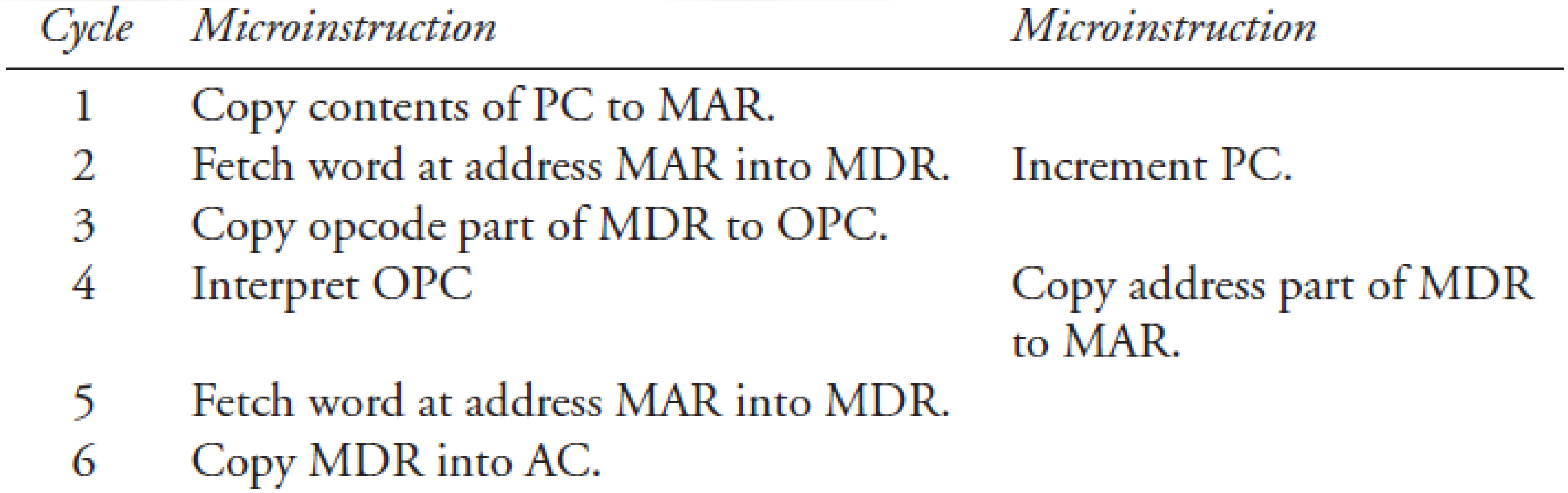
outputs to the environment

The Control Unit is a State Machine



The Fetch-and-Execute Cycle

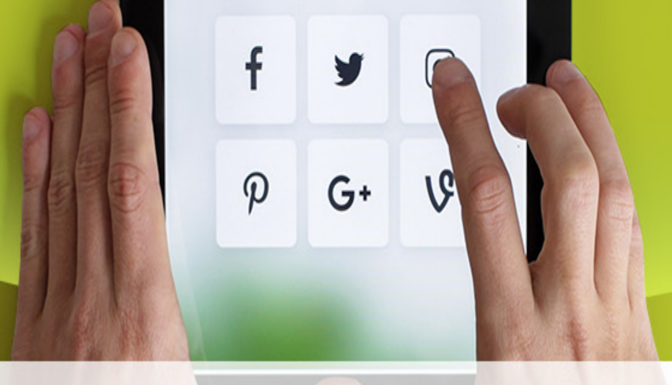
- Suppose OPC is a **load accumulator** instruction.
- the action required is to copy the word specified by the address part of the instruction into the accumulator
- **load accumulator** is decomposed into 8 **microinstructions**



Cycle	*Microinstruction*	*Microinstruction*
1	Copy contents of PC to MAR.	
2	Fetch word at address MAR into MDR.	Increment PC.
3	Copy opcode part of MDR to OPC.	
4	Interpret OPC	Copy address part of MDR to MAR.
5	Fetch word at address MAR into MDR.	
6	Copy MDR into AC.	

System Clock

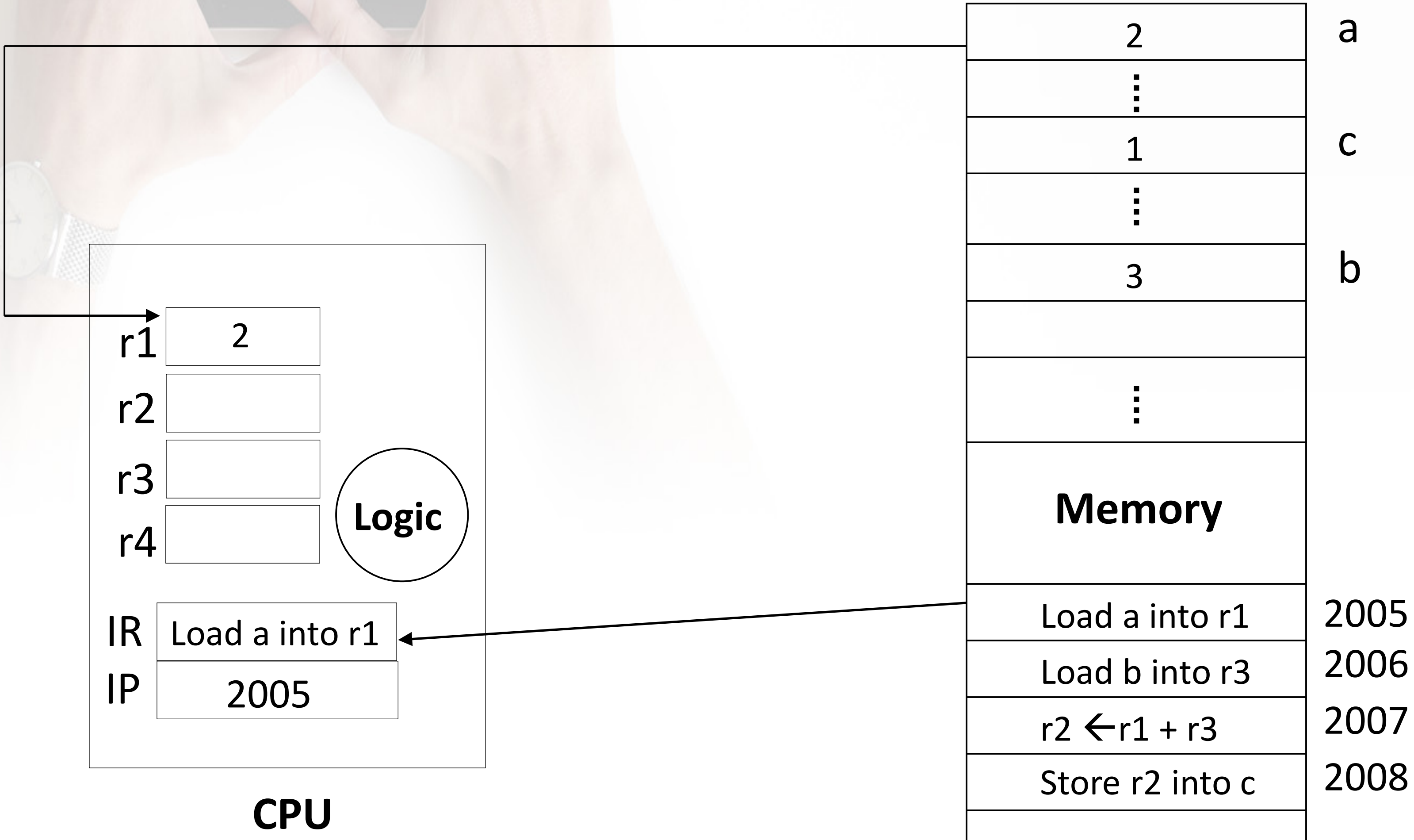
- System clock produces pulses at a fixed rate
- Each pulse is one Machine Cycle
- One program instruction may actually be several instructions to the CPU
- Each CPU instruction will take one pulse
- CPU has an instruction set – instructions that it can understand and process



A Simple Program

- Want to add values of variables a and b (assumed to be in memory), and put the result in variable c in memory, I.e. c ← a+b
- Instructions in program
  - Load a into register r1
  - Load b into register r3
  - r2 ← r1 + r3
  - Store r2 in c

Running the Program

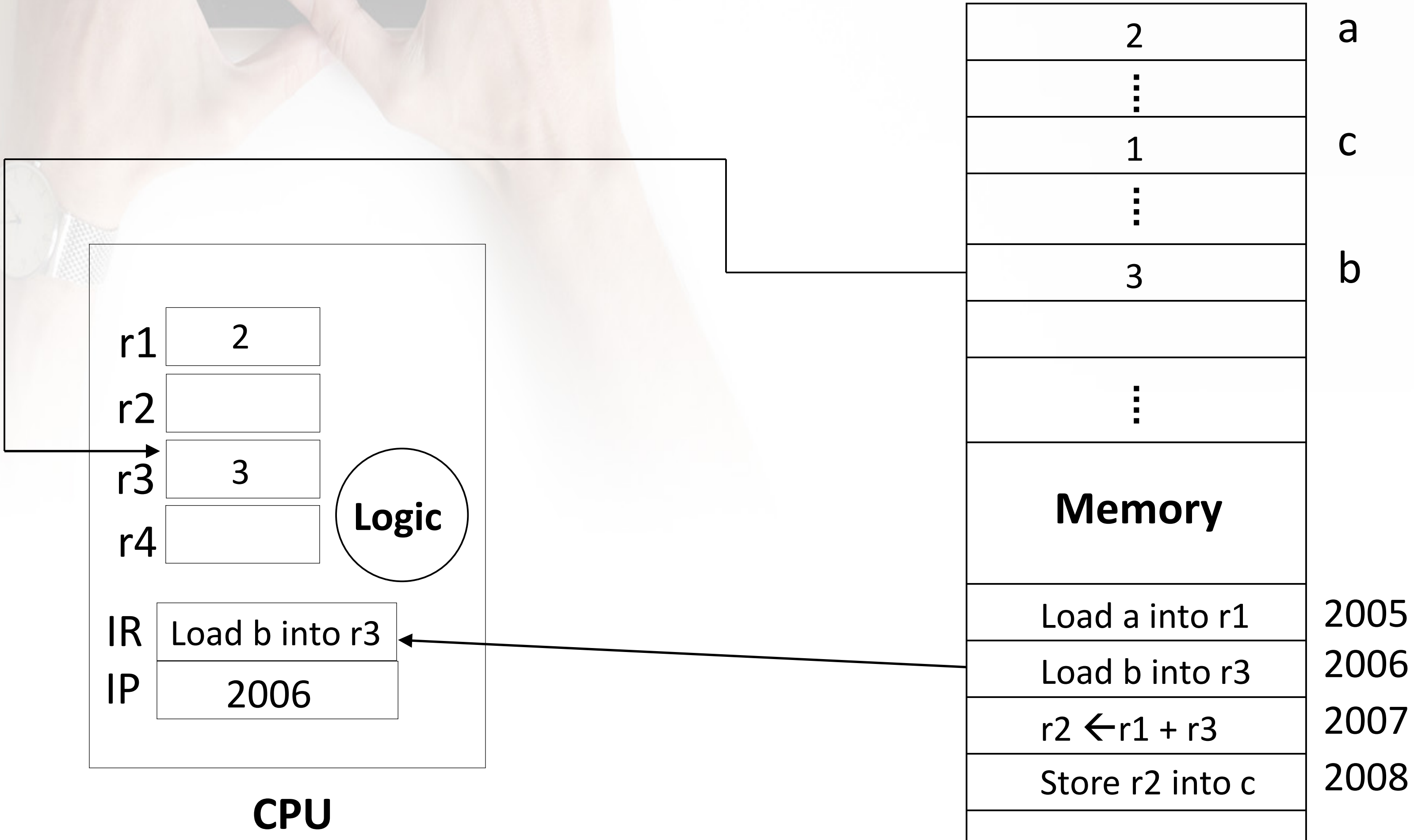


2	a
⋮	
1	c
⋮	
3	b
⋮	
Memory	
Load a into r1	2005
Load b into r3	2006
r2 ←r1 + r3	2007
Store r2 into c	2008

r1	2
r2	
r3	
r4	
IR	Load a into r1
IP	2005

CPU

Running the Program



r1	2
r2	
r3	3
r4	

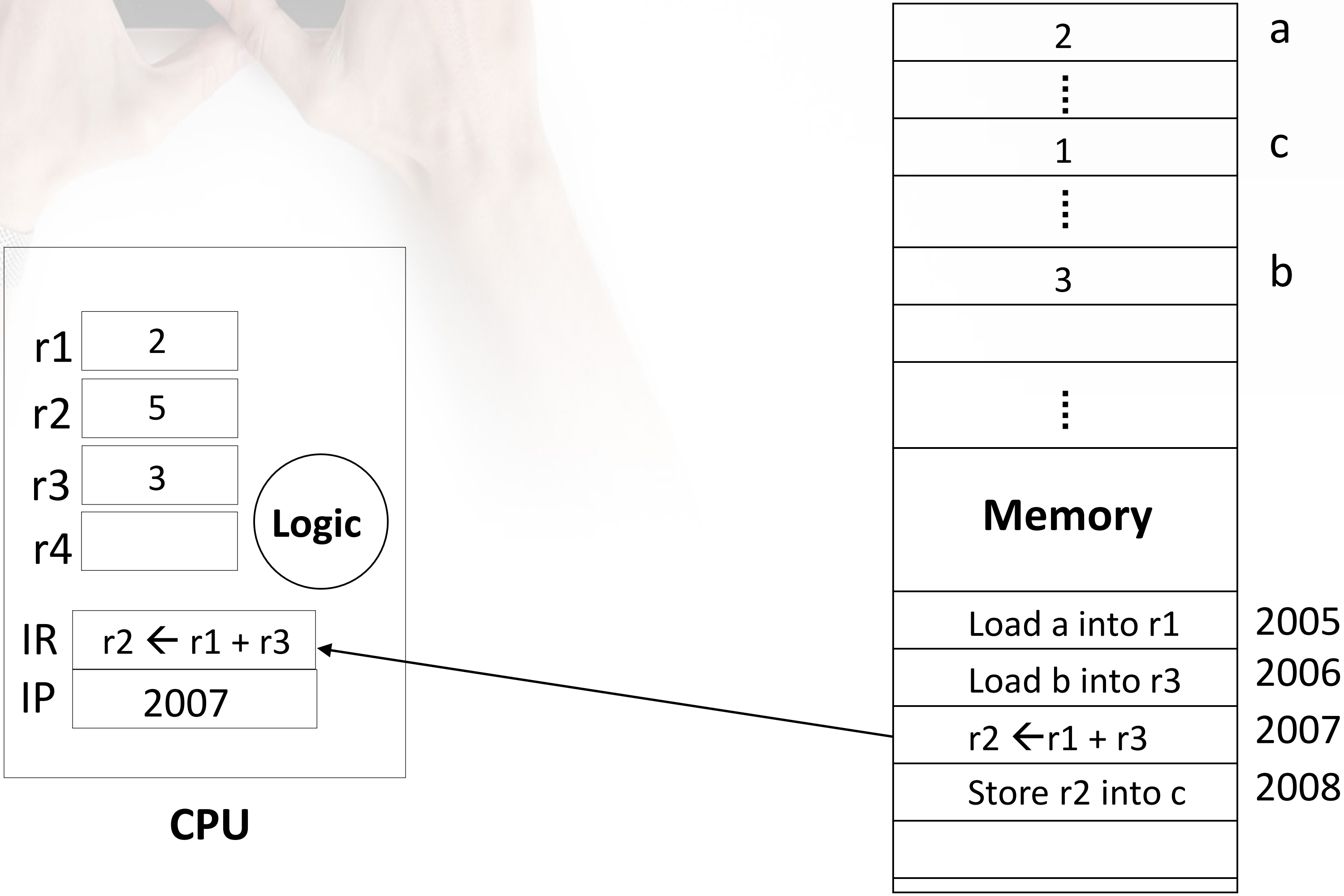
Logic

IR	Load b into r3
IP	2006

CPU

2	a
⋮	
1	c
⋮	
3	b
⋮	
Memory	
Load a into r1	2005
Load b into r3	2006
r2 ←r1 + r3	2007
Store r2 into c	2008

Running the Program



r1 2

r2 5

r3 3

r4

Logic

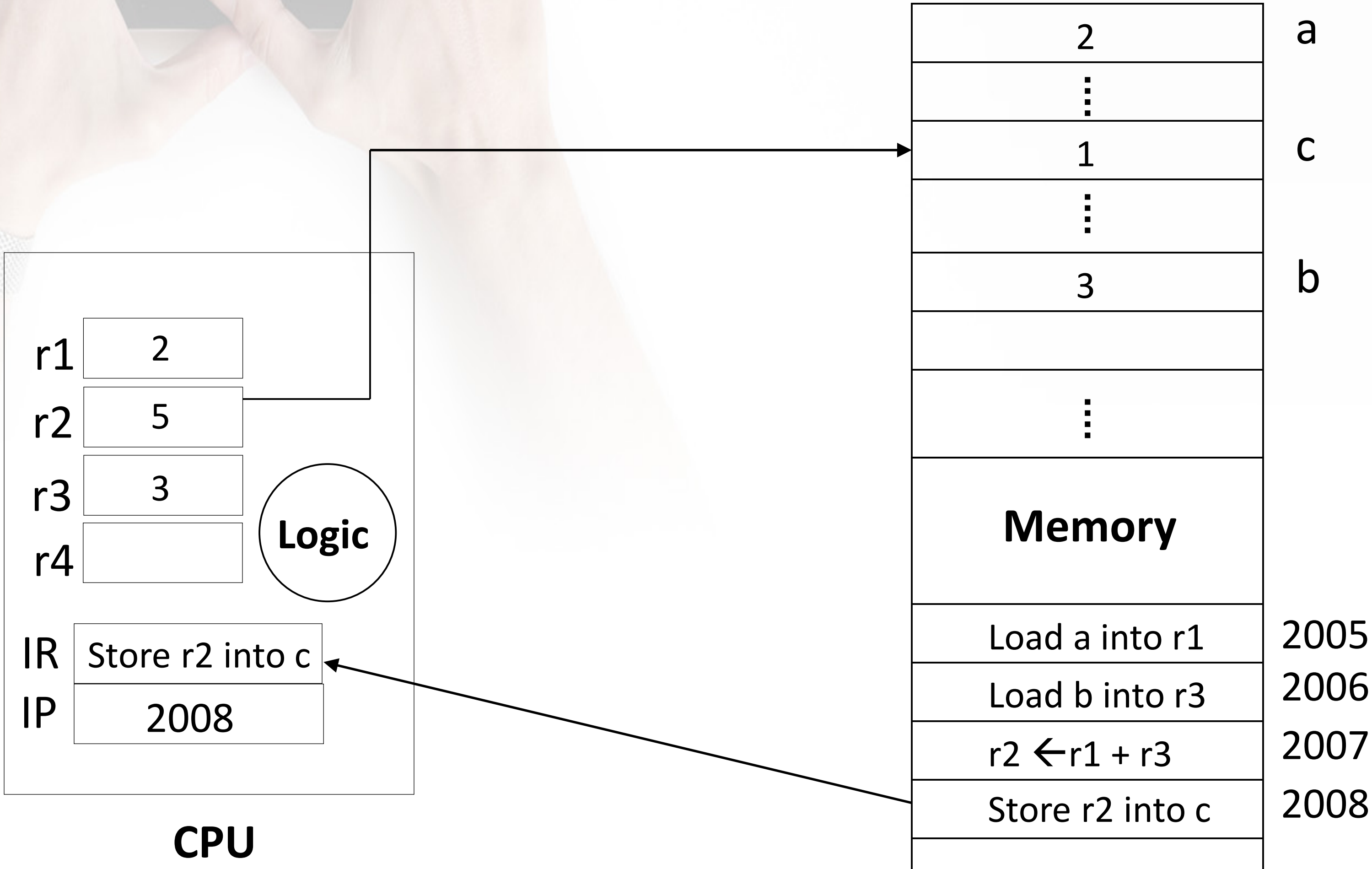
IR r2 ← r1 + r3

IP 2007

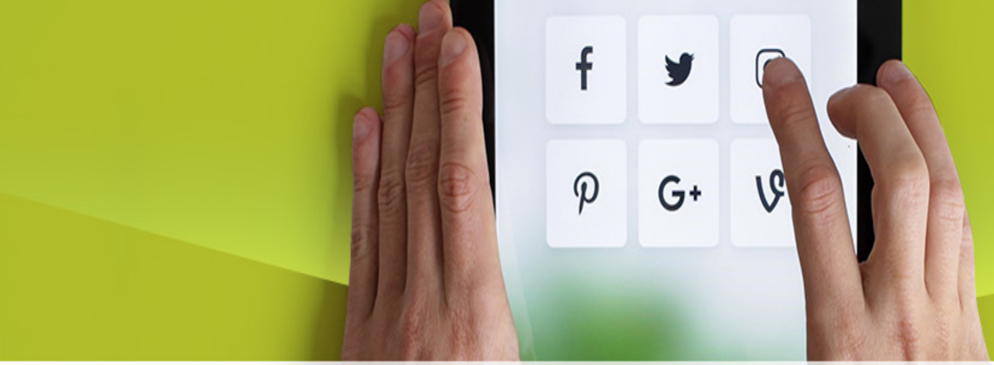
CPU

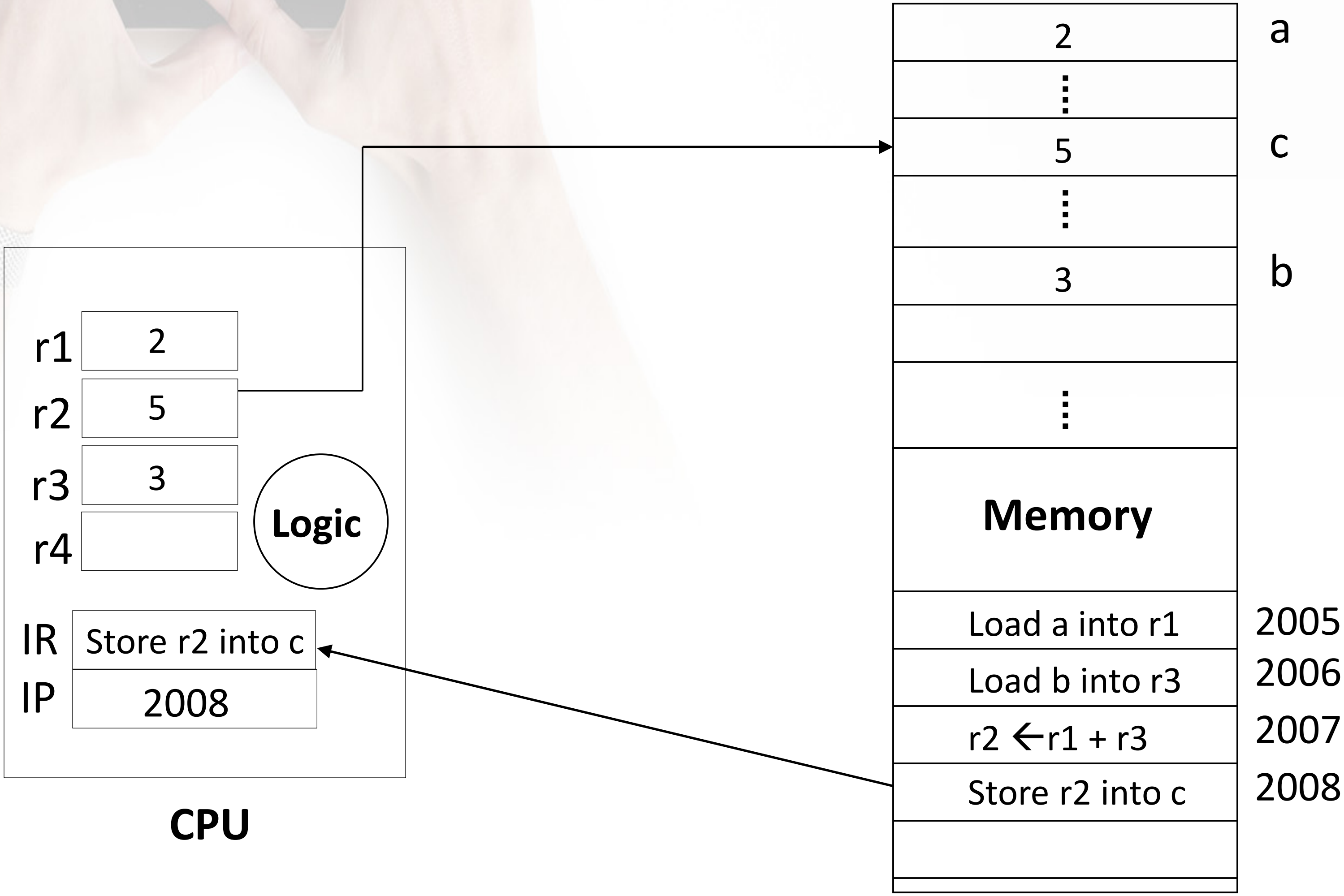
2	a	
⋮		
1	c	
⋮		
3	b	
⋮		
Memory		
Load a into r1	2005	
Load b into r3	2006	
r2 ←r1 + r3	2007	
Store r2 into c	2008	

Running the Program



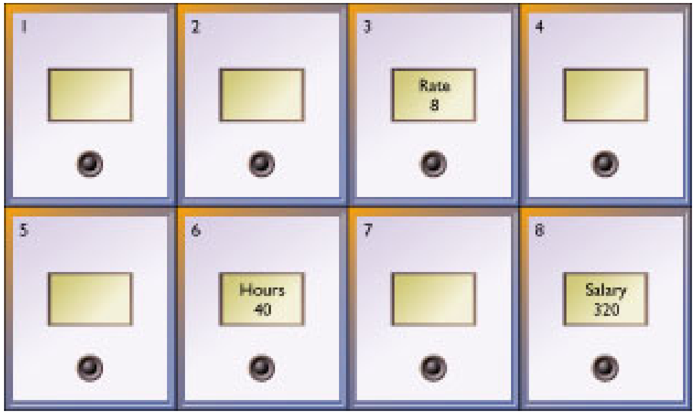
Running the Program





Finding Data in Memory

- Each location in memory has a unique address
  - Address never changes
  - Contents may change
- Memory location can hold one instruction or piece of data
- Programmers use symbolic names



Speed and Power

What makes a computer fast?

- Microprocessor speed
- Bus line size
- Availability of cache
- Flash memory
- RISC computers
- Parallel processing

Computer Processing Speed

Time to execute an instruction

- Millisecond
- Microsecond
- Nanosecond
  - Modern computers
- Picosecond
  - In the future

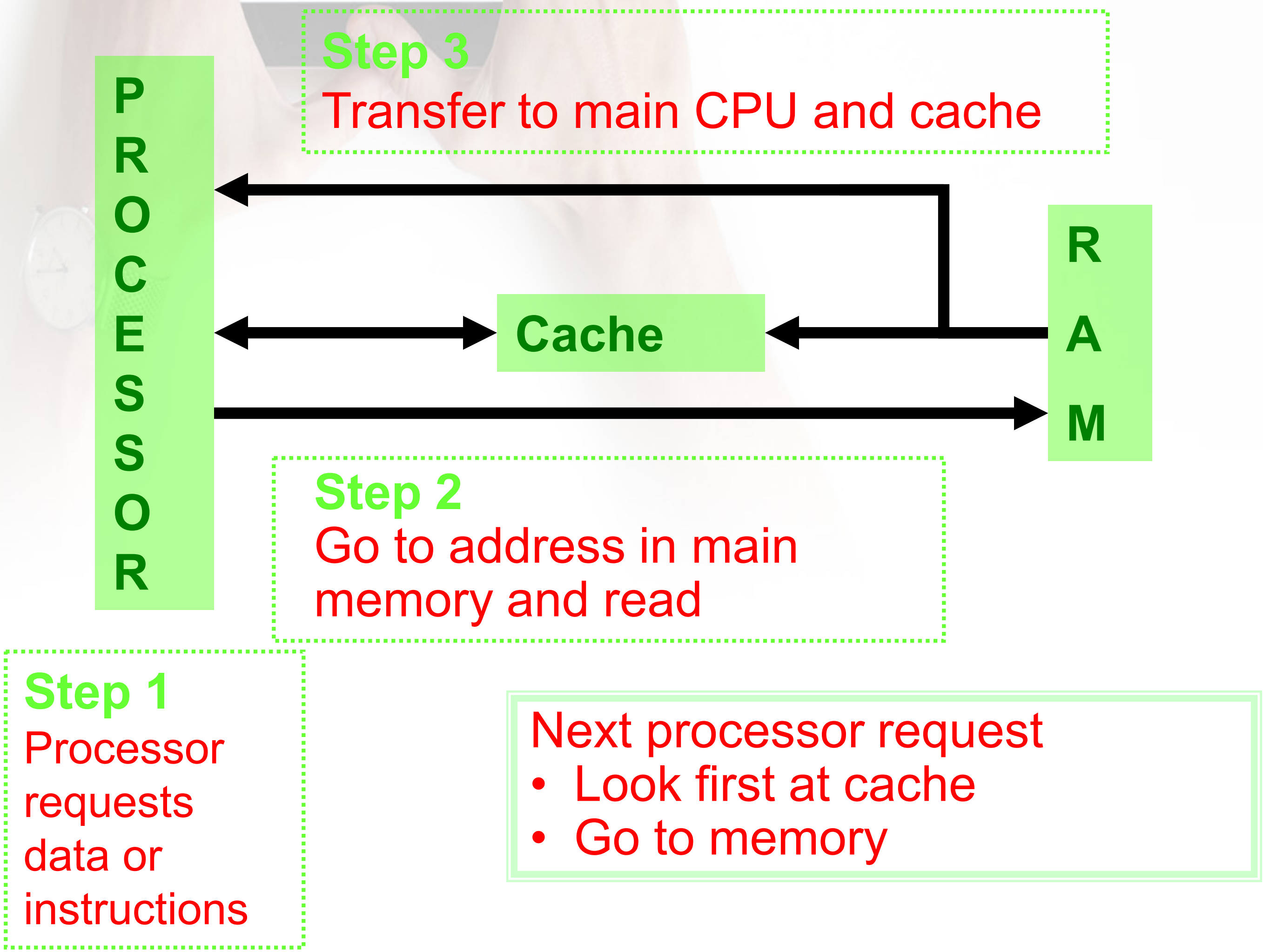
Microprocessor Speed

- Clock speed
  - Megahertz (MHz)
  - Gigahertz (GHz)
- Number of instructions per second
  - Millions of Instructions Per Second (MIPS)
- Performance of complex mathematical operations
  - One million floating-point operations per second (Megaflop)

Cache

- Small block of very fast temporary memory
- Speed up data transfer
- Instructions and data used most frequently or most recently

Cache



Step 3
Transfer to main CPU and cache

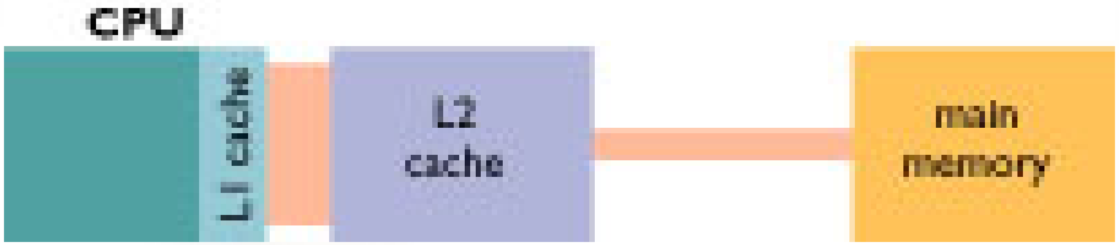
Step 2
Go to address in main memory and read

Step 1
Processor requests data or instructions

Next processor request
- Look first at cache
- Go to memory

Types of Cache

- Internal cache
  - Level 1 (L1)
  - Built into microprocessor
  - Up to 128KB
- External cache
  - Level 2 (L2)
  - Separate chips
  - 256KB or 512 KB
  - SRAM technology
  - Cheaper and slower than L1
  - Faster and more expensive than memory

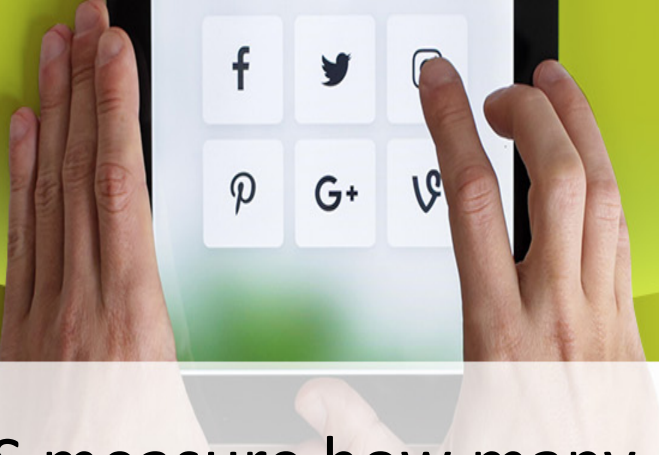


what is a TFLOP?

- TFLOP is a bit of shorthand for “teraflop,” which is a way of measuring the power of a computer based more on mathematical capability than GHz. A teraflop refers to the capability of a processor to calculate one trillion floating-point operations per second.
- Saying something has “6 TFLOPS,” for example, means that its processor setup is capable of handling 6 trillion floating-point calculations every second, on average.

What are floating-point calculations?

- From a computational standpoint, a floating-point calculation is any *finite* calculation that uses floating-point numbers, particularly decimals.
- This is far more useful than looking at fixed-point calculations (which use only whole integers), because the work that computers do frequently involves finite floating-point numbers and all their real world complications.



FLOPS

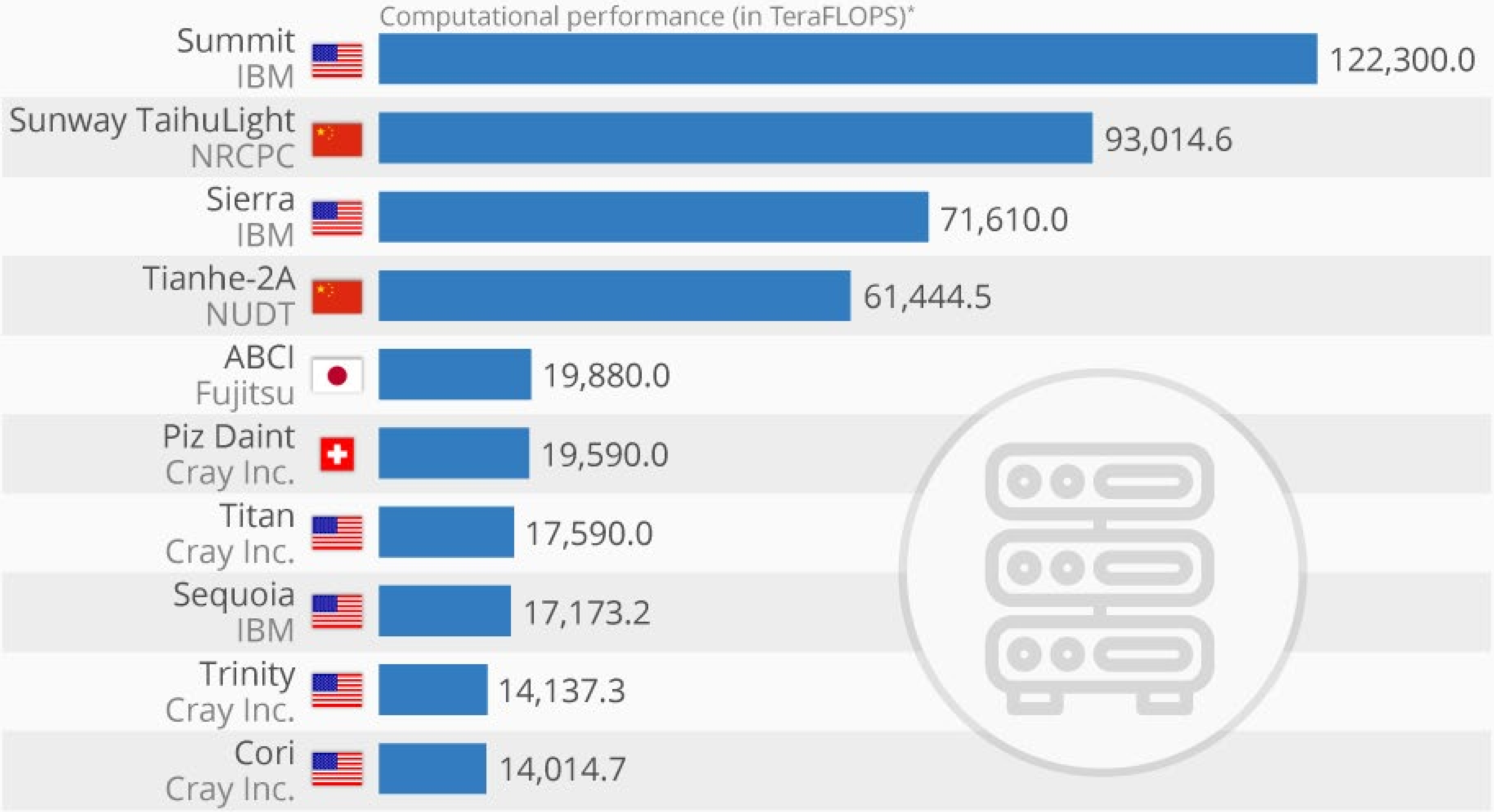
- FLOPS measure how many equations involving floating-point numbers that a processor can solve in one second.
- A traditional calculator, for example, may need only around 10 FLOPS for all its operations.
- So when we start talking about megaflops (a million floating-point calculations), gigaflops (a billion) and teraflops (a trillion), you can start to see what sort of power we're talking about.

The Top 10 Supercomputers

- The world’s fastest “supercomputer“ has a computing power of 122,300 teraflops, i.e. it can carry out 122,000 trillion floating point operations per second.
- While that only sounds like Chinese to the uninformed ear, Summit, the computer topping the current TOP500 ranking of supercomputers, is not located in China.
- Even though the far eastern country is one of the most important hubs for supercomputing, the United States, its direct competitor in the never-ending run for the world’s most powerful calculator, is currently home of the aforementioned supercomputer which was revealed at the Oak Ridge National Laboratory yesterday.
- Supercomputers are used to run complicated simulations that involve a large number of variables. Common use cases include economic and climate modeling, neurological research and nuclear science.

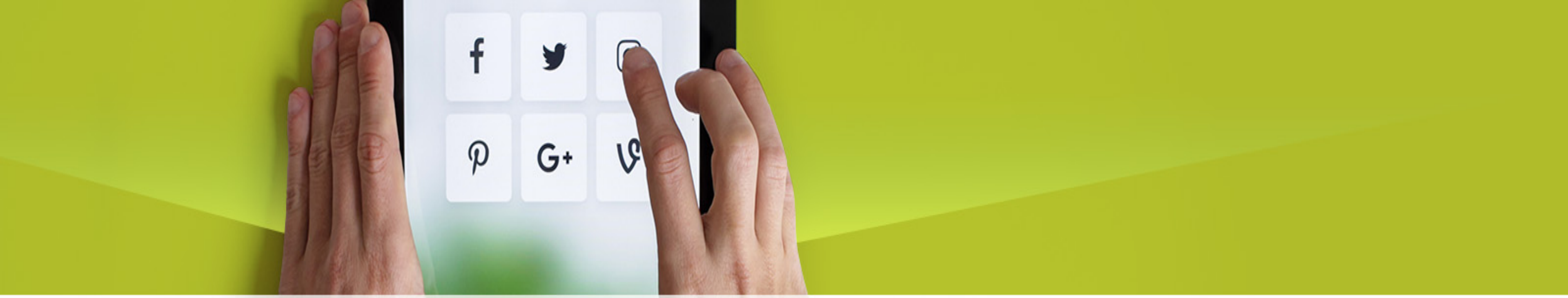
The Top 10 Supercomputers

Computational performance of the most powerful Supercomputers (as of June 2018)



* FLOPS = floating point operations per second, i.e. the number of basic mathematical operations a computer can perform in a second

Source: top500.org

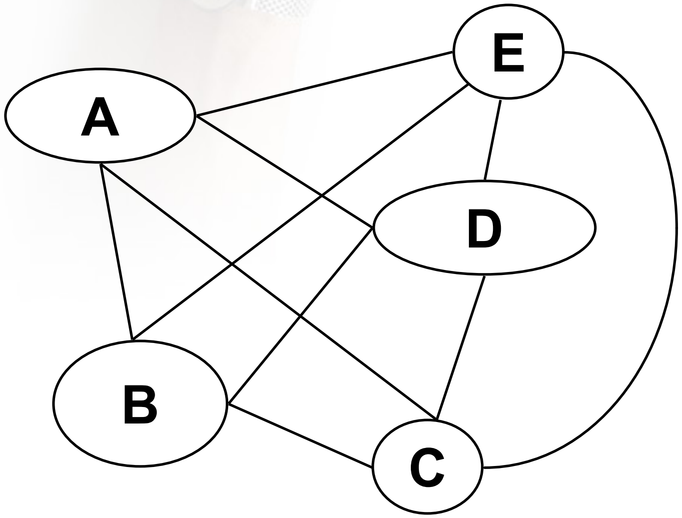


Lecture 2 Chapter 2

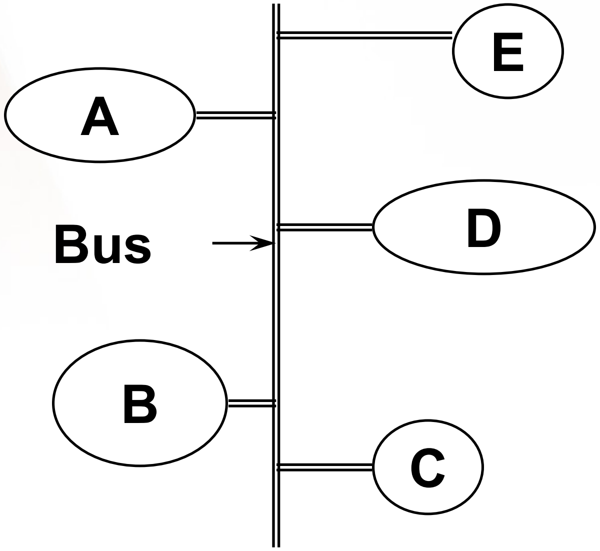
BUSES

Buses

- Concept is to link together multiple functional units over a common data highway at a lower cost than using multiple point to point links



OR



Number of Links = n * (n – 1) / 2

Computer Buses

- A **bus** is a common electrical pathway between multiple devices.
  - Can be internal to the CPU to transport data to and from the ALU.
  - Can be external to the CPU, to connect it to memory or to I/O devices.
- Early PCs had a single external bus or **system bus**.
- Modern PCs have a special-purpose bus between the CPU and memory and (at least) one other bus for the I/O devices.

Bus Line

- Paths that transport electrical signals
- System bus
  - Transports data between the CPU and memory
- Bus width
  - Number of bits of data that can be carried at a time
  - Normally the same as the CPUs word size
- Speed measured in MHz

Bus Line

Larger bus width	=	More powerful computer
CPU can transfer more data at a time	=	Faster computer
CPU can reference larger memory addresses	=	More memory available

CPU can support a greater number and variety of instructions

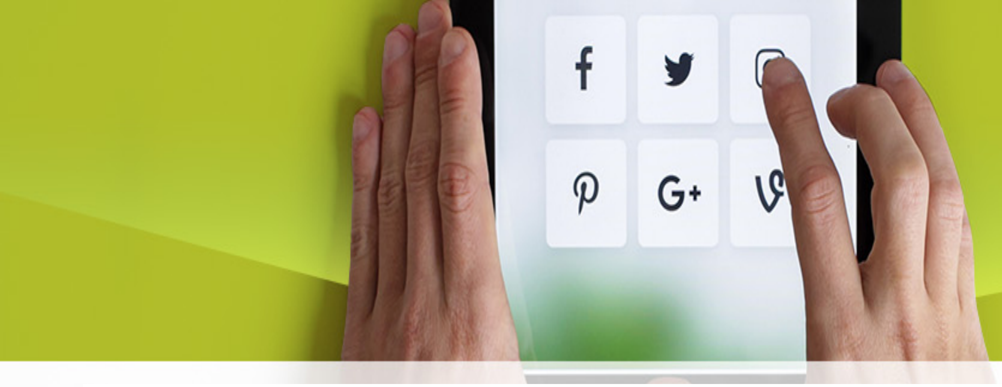
PC Buses and Ports

ISA	Slow-speed devices like mouse, modem
PCI	High-speed devices like hard disks and network cards
AGP	Connects memory and graphics card for faster video performance
USB	Supports “daisy-chaining” eliminating the need for multiple expansion cards; hot-swappable
IEEE 1394 (FireWire)	High-speed bus connecting video equipment to the computer
PC Card	Credit card sized PC card devices normally found on laptops

Computer Buses

- A number of buses are in widespread use in the computer world.
  - Multibus (8086)
  - IBM PC (PC/XT)
  - ISA bus (PC/AT)
  - EISA bus (80386)
  - Microchannel (PS/2)
  - PCI bus (Many PCs)
  - Nubus (macintosh)
  - Universal Serial Bus (modern PCs)
  - FireWire (consumer electronics)

Computer Buses



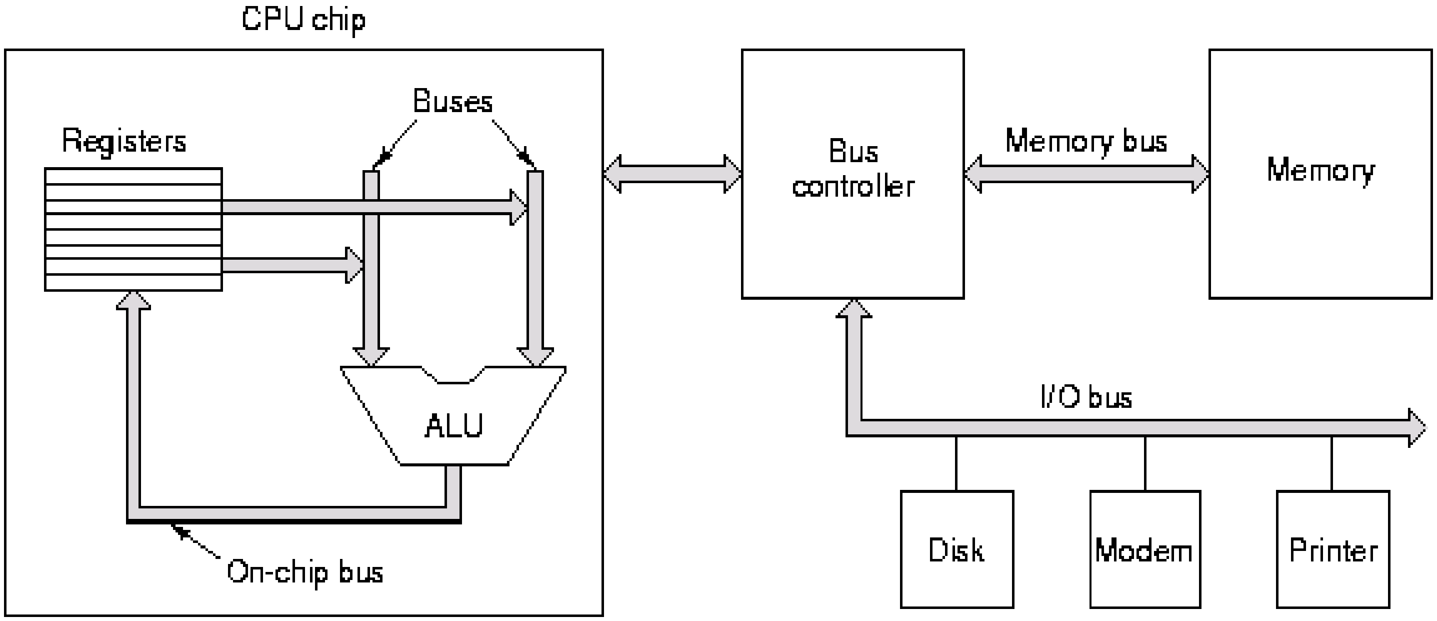
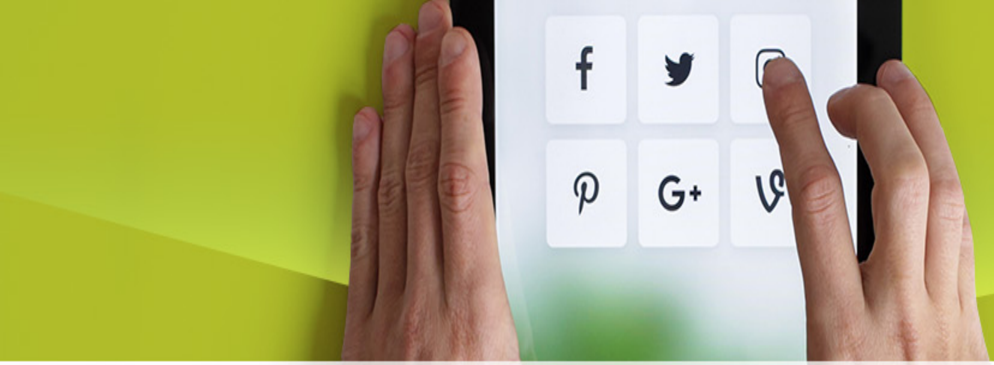
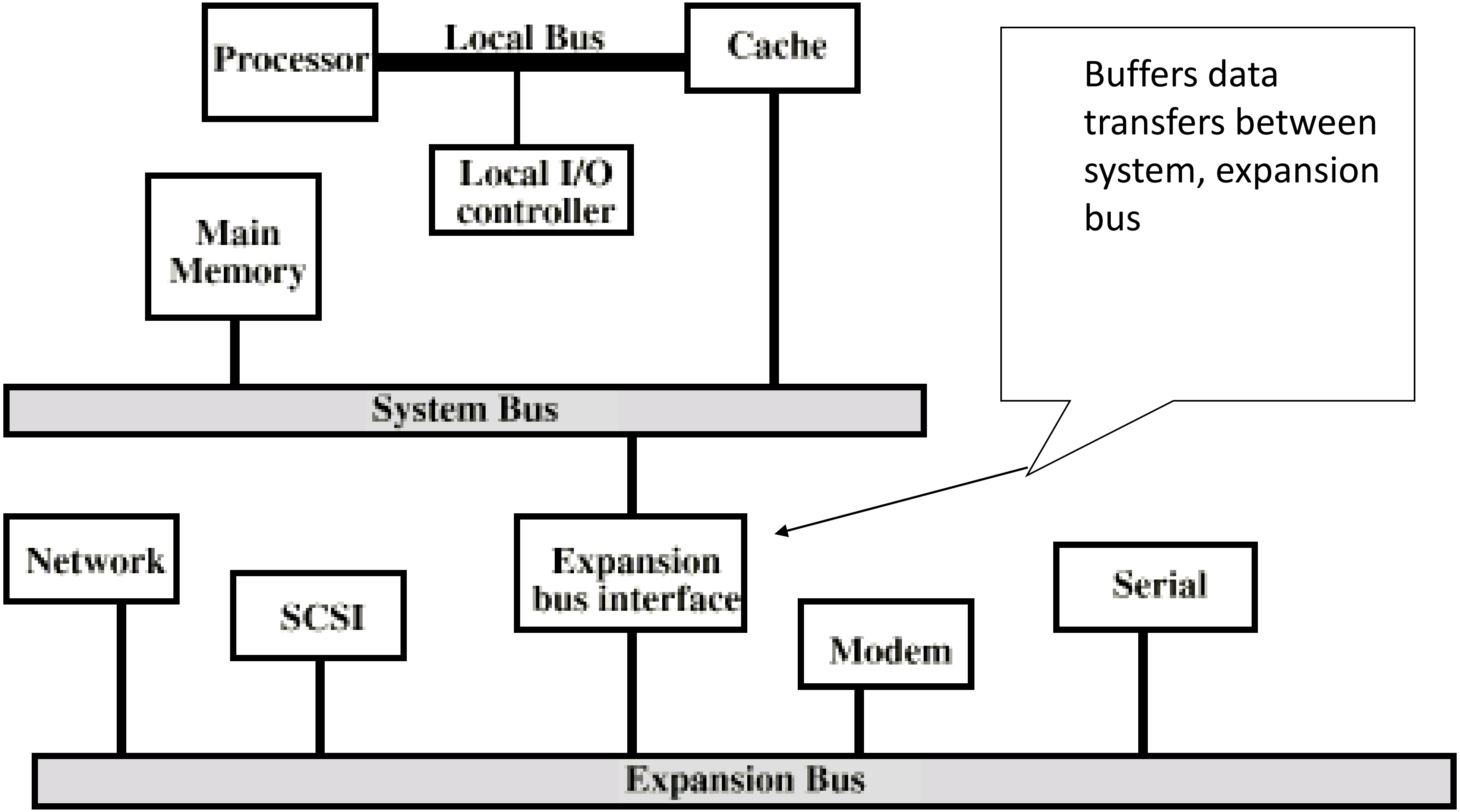


Figure 3-34. A computer system with multiple buses.



Traditional (ISA) (with cache)



Buffers data transfers between system, expansion bus

This approach breaks down as I/O devices need higher performance

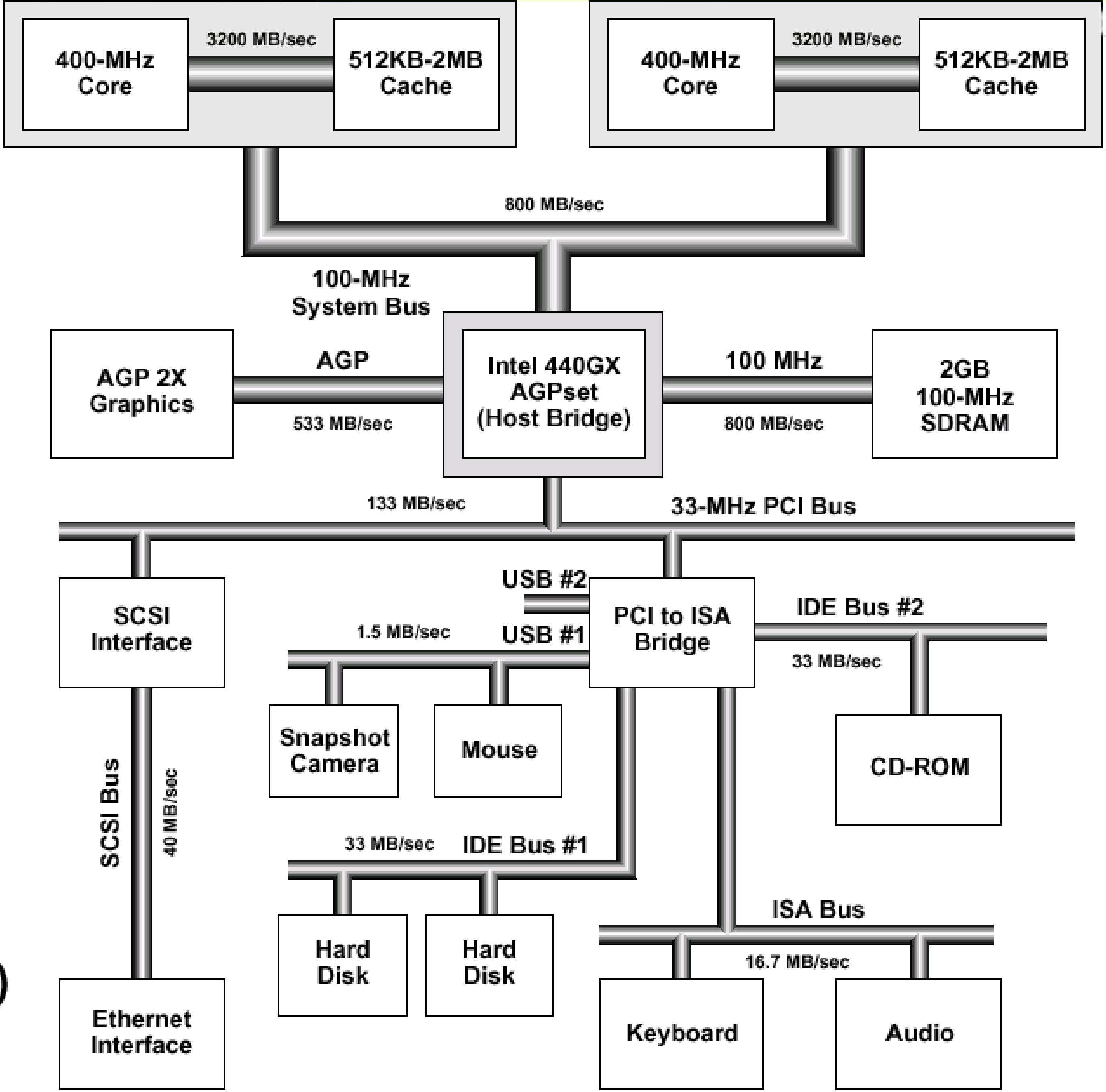
Computer Buses

- In order to make it possible for boards designed by third parties to attach to the system bus, there must be well-defined rules about how the bus works, and which all attached devices must obey.
- These rules are called the **bus protocol**.
- In addition, there must be mechanical and electrical specifications.

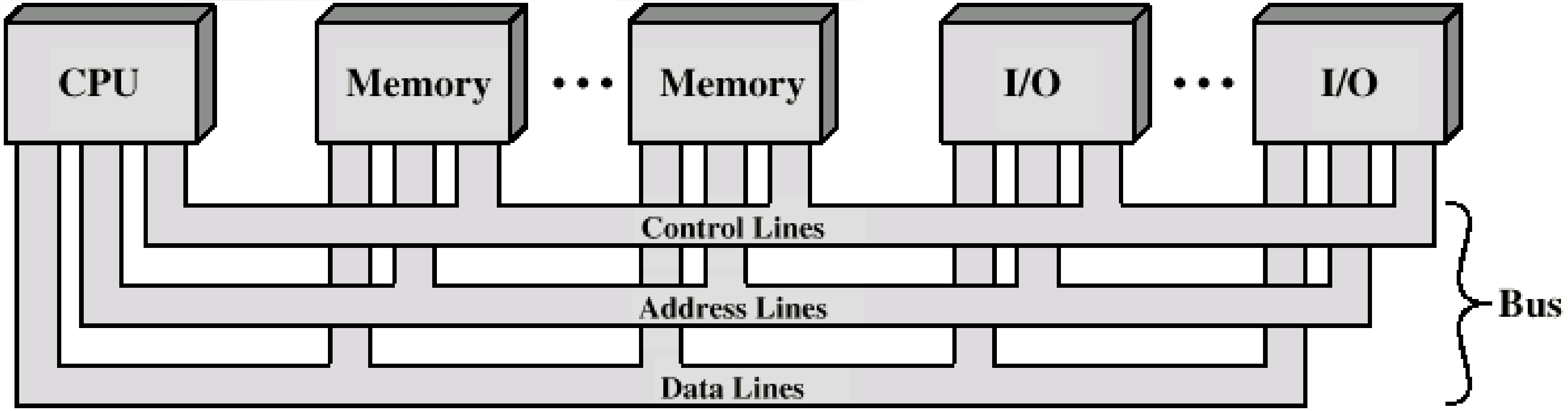
Bridge Based Bus Architecture

- Bridging with dual Pentium II Xeon processors on Slot 2.

(Source: http://www.intel.com.)



Bus Interconnection Scheme



Data Bus

- Carries data
  - Remember that there is no difference between “data” and “instruction” at this level
- Width is a key determinant of performance
  - 8, 16, 32, 64 bit
  - What if the data bus is 8 bits wide but instructions are 16 bits long?
  - What if the data bus is 64 bits wide but instructions are 16 bits long?

Address bus

- Identify the source or destination of data
  - In general, the address specifies a specific memory address or a specific I/O port
- e.g. CPU needs to read an instruction (data) from a given location in memory
- Bus width determines maximum memory capacity of system
  - 8086 has 20 bit address bus but 16 bit word size for 64k directly addressable address space
  - But it could address up to 1MB using a segmented memory model
    - RAM: 0 – BFFFF, ROM: C0000 - FFFFF
    - DOS only allowed first 640K to be used, remaining memory for BIOS, hardware controllers. Needed High-Memory Manager to “break the 640K barrier”

Control Bus

- Control and timing information
  - Determines what modules can use the data and address lines
  - If a module wants to send data, it must (1) obtain permission to use the bus, and (2) transfer data – which might be a request for another module to send data
- Typical control lines
  - Memory read
  - Memory write
  - I/O read
  - I/O write
  - Interrupt request
  - Interrupt ACK
  - Bus Request
  - Bus Grant
  - Clock signals

8088 / 8086 CPU in Min Mode

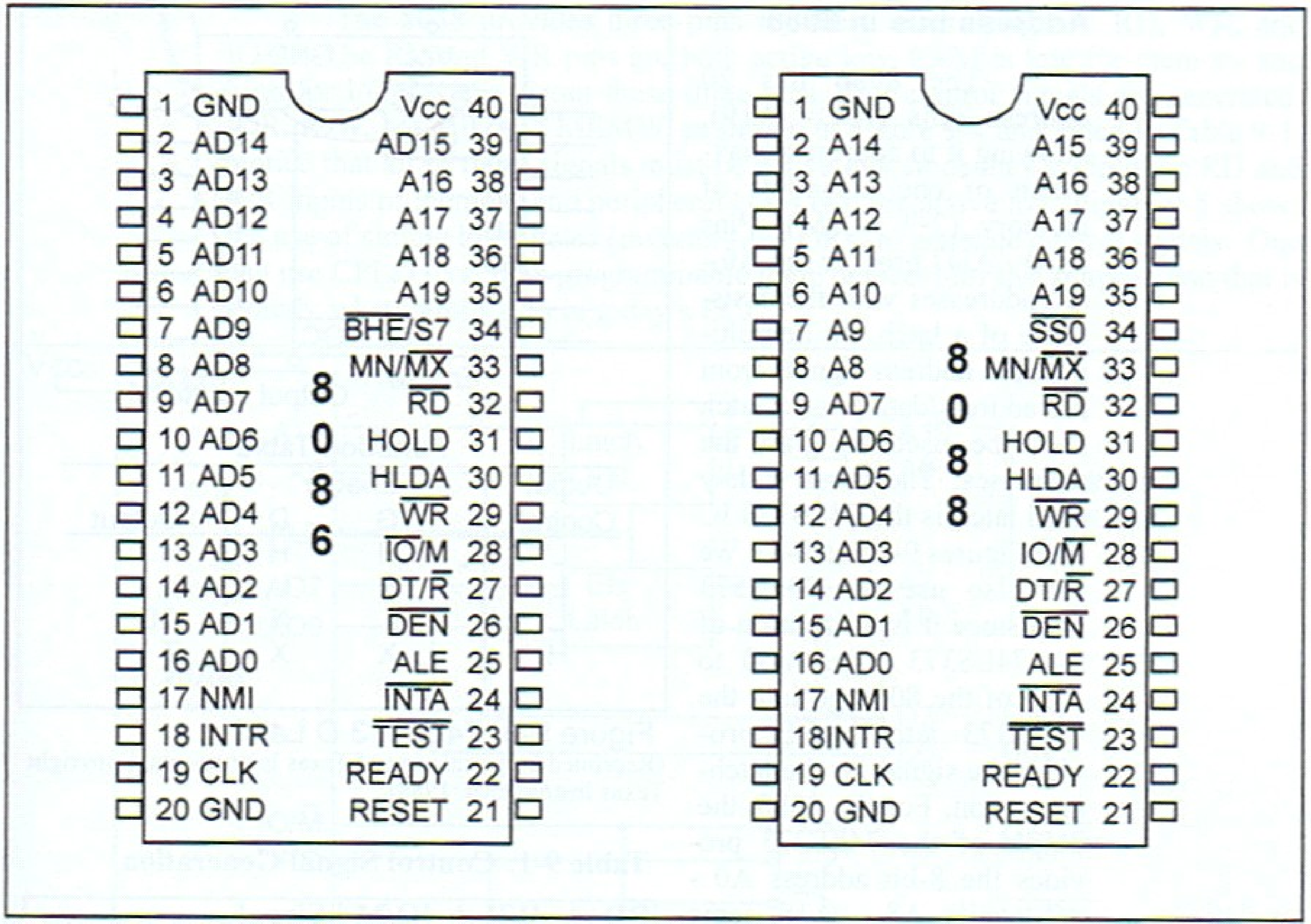


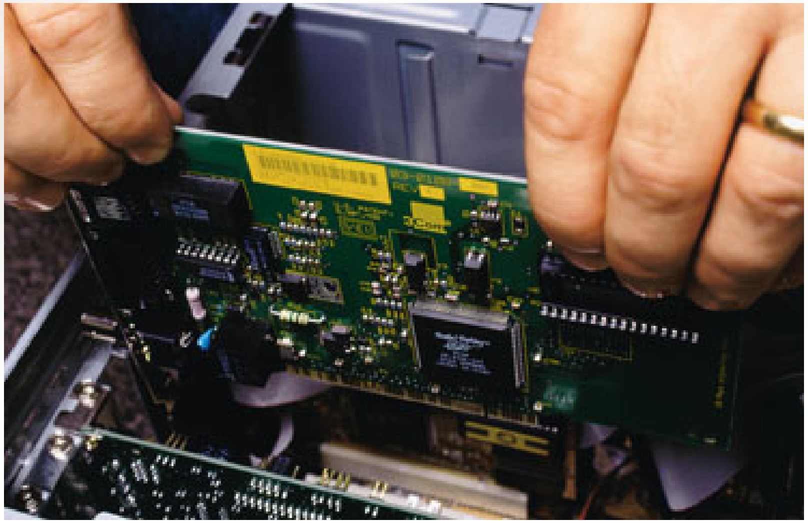
Figure 9-1. The 8086 and 8088 in Minimum Mode

(Reprinted by permission of Intel Corporation, Copyright Intel, 1989)

Expansion Buses

- Connect the motherboard to expansion slots
- Plug expansion boards into slots
  - interface cards
  - adapter cards
- Provides for external connectors / ports
  - Serial
  - Parallel

Expansion Buses



Computer Buses

- Some devices that attach to a bus are active and can initiate bus transfers. They are called **masters**.
- Some devices are passive and wait for requests. They are called **slaves**.
- Some devices may act as slaves at some times and masters at others.
- Memory can never be a master device.

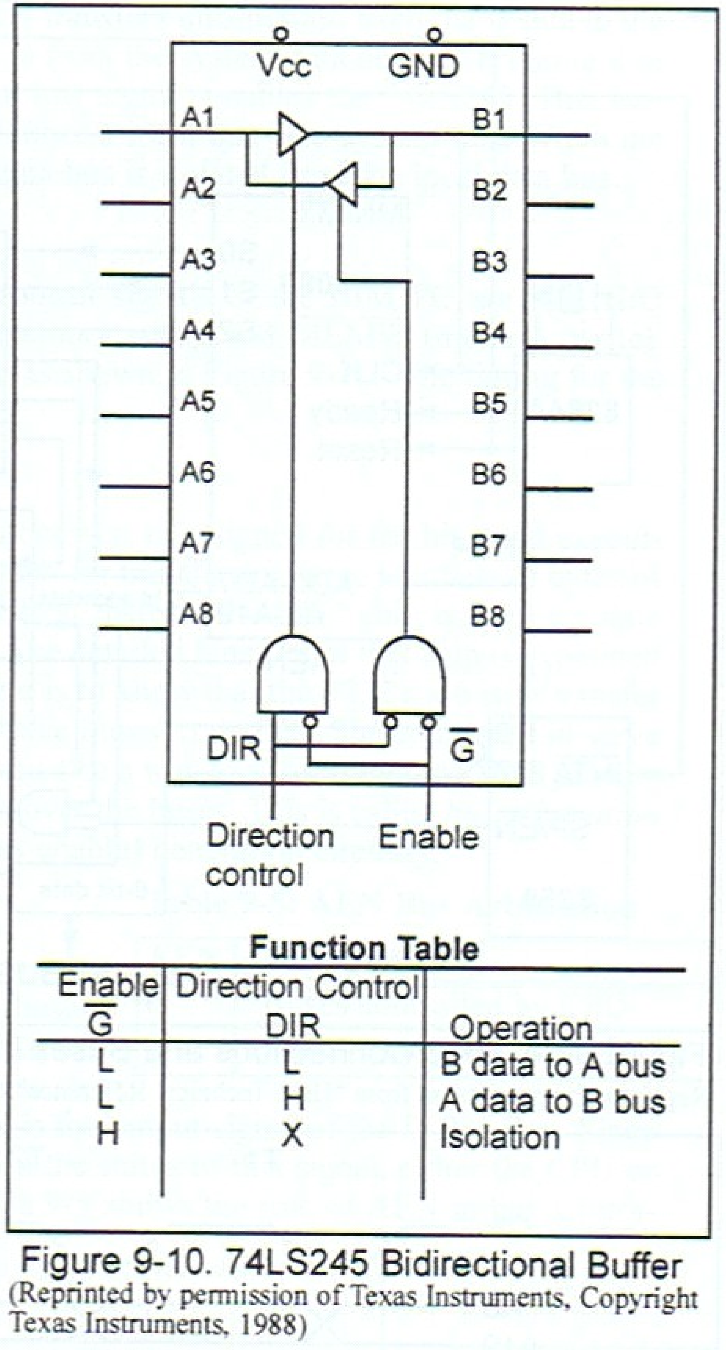
Computer Buses

- The binary signals that computer devices output are frequently not strong enough to power a chip.
  - The bus may be relatively long or have several devices attached to it.
  - Most bus masters are connected to the bus by a chip called a **bus driver** which is essentially a digital amplifier.
  - Most slaves are connected to the bus by a **bus receiver**.

Computer Buses

- For devices which can be both master and slave, a device called a **bus transceiver** is used.
- These bus interface devices are often tri-state devices to allow them to disconnect when they are not needed.
- A bus has address, data, and control lines, but there is not necessarily a one-to-one mapping between CPU pins and bus lines. A decoder chip between CPU and bus would be needed in this case.

Bidirectional Buffer (74LS245)



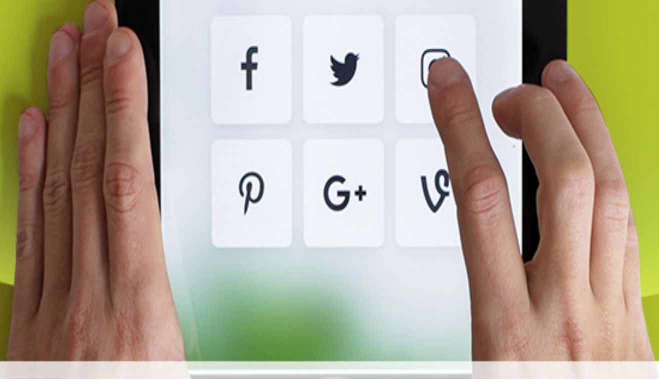
Function Table

Enable $\overline{G}$	Direction Control DIR	Operation
L	L	B data to A bus
L	H	A data to B bus
H	X	Isolation

Figure 9-10. 74LS245 Bidirectional Buffer
(Reprinted by permission of Texas Instruments, Copyright Texas Instruments, 1988)

Bus Width

- – The more address lines a bus has, the more memory the CPU can address directly.
- – If a bus has *n* address lines, then the CPU can use it to address 2^n different memory locations.
- – Larger buses are more expensive:
  - they need more wires
  - they take up more space on the motherboard
  - they need bigger connectors
  - Early PC buses did not contain enough address lines leading several backward compatible upgrades to the bus.



Bus Width

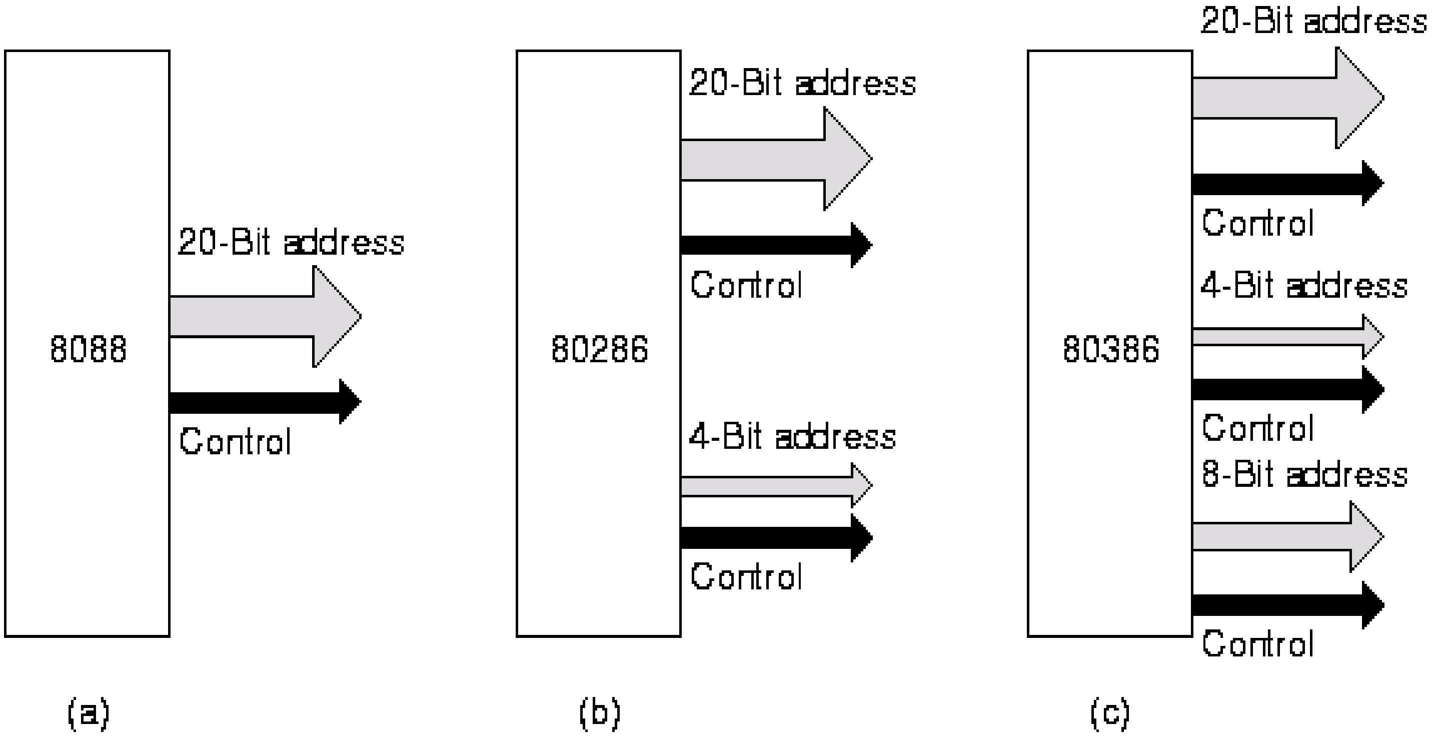


Figure 3-36. Growth of an address bus over time.

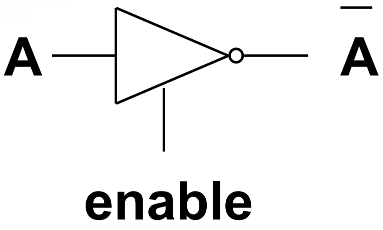
Tri-state Logic Outputs

- Since we can have multiple masters on a bus, we need Tri-state logic for attachment to a bus so that each device can choose to drive or not drive the bus depending on whether it is the bus master for a given bus cycle
- Tri-state logic prevents a bus conflict where one device is driving a signal to 1 and another device is driving it to 0 at the same time - generates high current through wires (and smoke?)

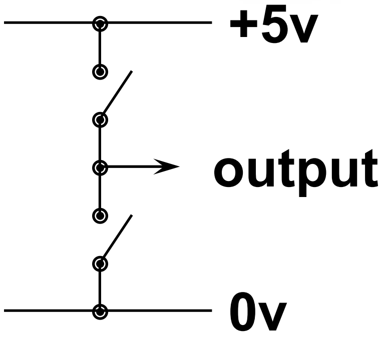
Tri-State Logic

- The problem with connecting multiple “normal” outputs together on a bus is that each has to be in one logic state (0) or the other (1) - driving voltage on each bus signal high or low
- This represents a conflict over the state of the signal
- We resolve this conflict with *tri-state logic*

Logically



Electrically

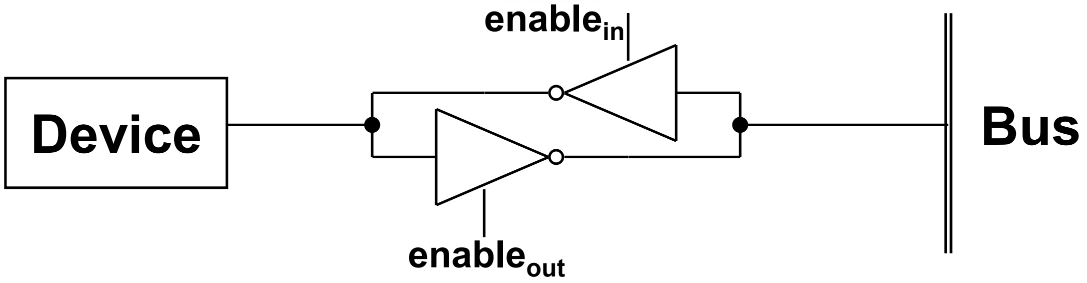


Truth Table

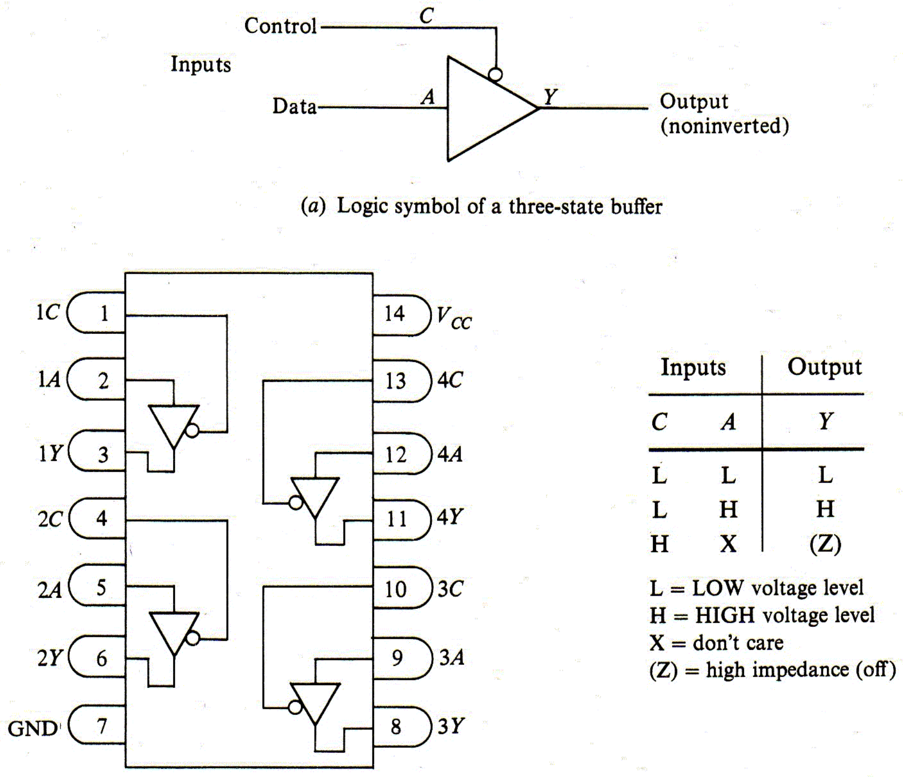
enable	A	Output
0	0	(Z)
0	1	(Z)
1	0	1
1	1	0

Tri-State Logic and Buses

- The logical element has output enable pin to go from a floating output to drive the output from the circuit
- Inverters and buffers are used as bus drivers or buffers
  - Two such drivers or buffers in opposite directions are used to make the connection bi-directional
  - The gates also provide more “drive” onto the bus so that the bus signals are stronger and the bus can be longer



Tri-State Logic and Buses



(*a*) Logic symbol of a three-state buffer

Inputs		Output
C	*A*	*Y*
L	L	L
L	H	H
H	X	(Z)

L = LOW voltage level
H = HIGH voltage level
X = don't care
(Z) = high impedance (off)

(*b*) Pin diagram

(*c*) Truth table

Fig. 13-12 74125 quad three-state buffer IC

Bus Width

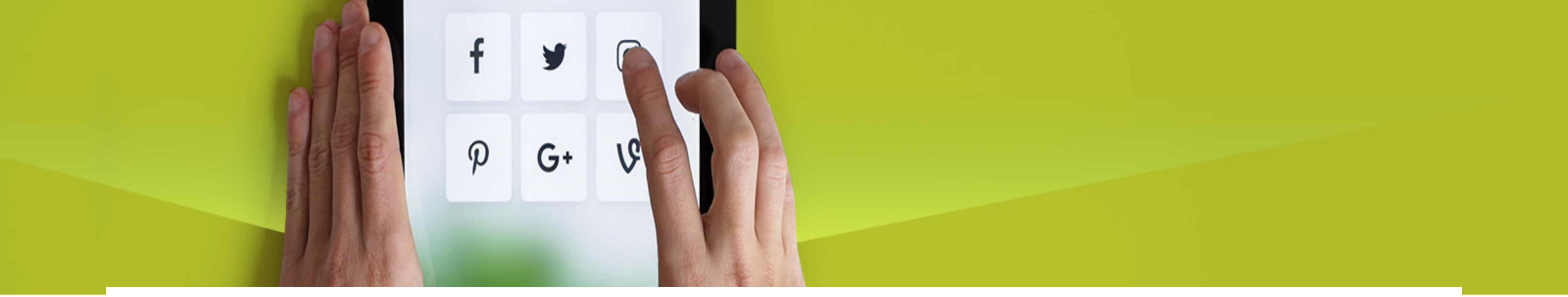
- – The number of data lines needed also tends to increase over time.
- – There are two ways to increase the data bandwidth of a bus:
  - decrease the bus cycle time
  - increase the data bus width
- – Speeding up the bus results in problems of bus skew since data on individual lines travel at slightly different speeds. This also makes the bus non-compatible with pre-existing devices.

Bus Width

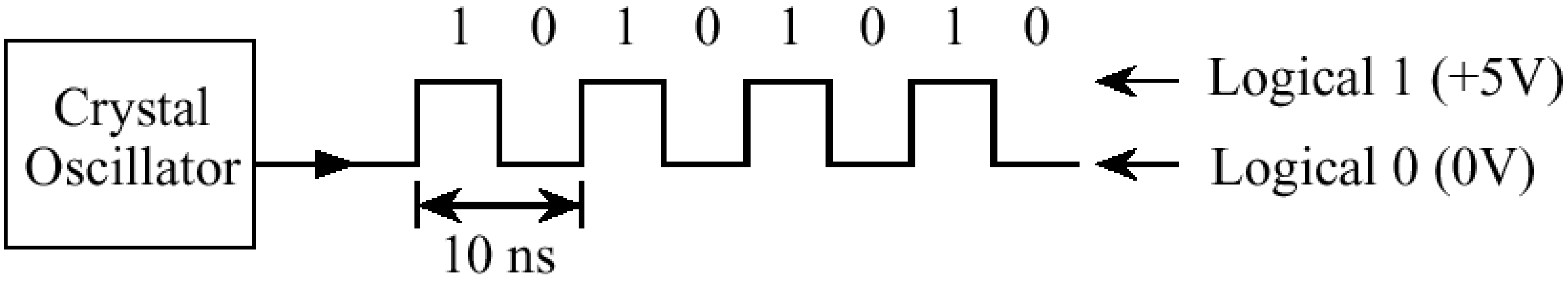
- Therefore, an increased data width is the usual answer (e.g. in the PC which went from 8 data lines to 16 and then to 32 on essentially the same bus).
- Another solution is to use a **multiplexed bus**.
- The same lines are used for both data and addressing by breaking up the bus operation into multiple steps. This slows down bus performance.

Timing

- Co-ordination of events on bus
- Synchronous
  - Events determined by clock signals
  - Control Bus includes clock line
  - A single 1-0 is a bus cycle
  - All devices can read clock line
  - Usually sync on leading edge
  - Usually a single cycle for an event

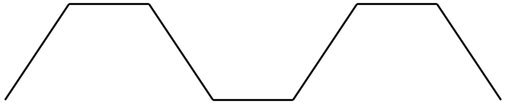


100 MHz Bus Clock



100 million cycles per second
1 cycle in (1/100,000,000) seconds = 0.0000001s = 10 ns

In reality, the clock is a bit more sawtoothed



Bus Clocking

- Buses can be divided up into two categories depending on their clocking.
- A **synchronous bus** has a line driven by a crystal oscillator.
  - The signal on this line consists of a square wave with a frequency of 5 - 100 MHz.
  - All bus activities take an integral number of these cycles, called **bus cycles**.
- The **asynchronous bus** does not have a master clock. Bus cycles can be of any length required and need not be the same.

Bus Clocking

    - This ends the read.
- A set of signals that interlocks in this way is called a **full handshake**.
- Full handshakes are timing independent. Each event is caused by a prior event, not by a clock cycle.
- Despite the advantages of asynchronous buses, most buses are synchronous since they are easier to build, and since there is such a large investment in synchronous bus technology.

Bus Clocking

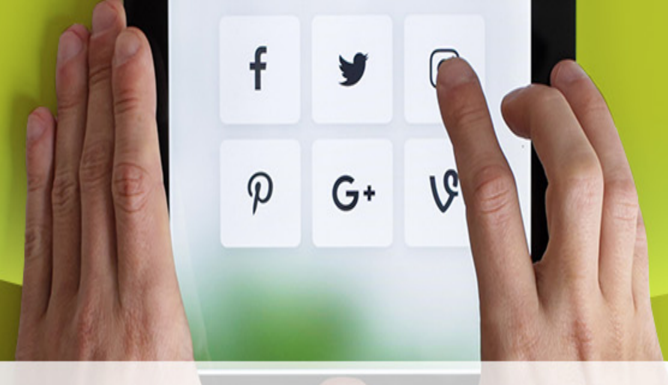
- Consider a synchronous bus with a 40-MHz clock, which gives a clock cycle of 25 nsec.
- Assume reading from memory takes 40 nsec from the time the address is stable.
  - It takes three bus cycles to read a word.
- MREQ’ indicates that memory is being accessed. RD’ is asserted for reads and negated for writes. WAIT’ inserts wait states (extra bus cycles) until the memory is finished

Bus Clocking

- Although synchronous buses are easy to work with due to their discrete time intervals, they also have some problems.
  - Everything works in multiples of the bus clock.
  - If a CPU and memory can complete a transfer in 3.1 cycles they have to stretch it to 4.0 because fractional cycles are forbidden.
  - Once a bus cycle has been chosen, and memory and I/O cards have been built for it, it is difficult to take advantage of future improvements in technology. The bus has to be geared to the slowest devices (legacy devices) on the bus.

Bus Clocking

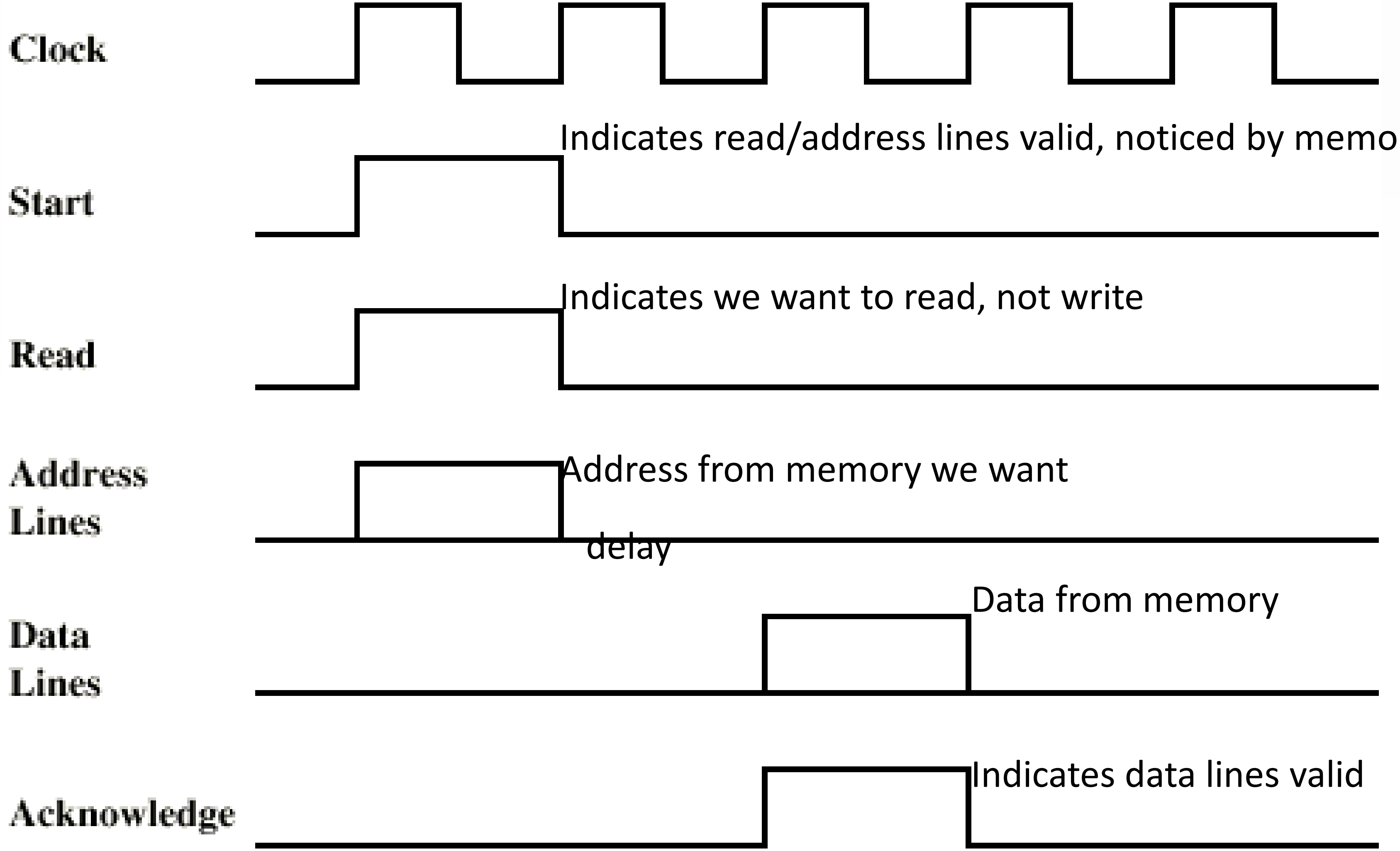
- Mixed technology can be handled by going to an asynchronous bus.
- The master device asserts MREQ’, RD’, etc. and then asserts MSYN’ (Master SYNchronization).
  - Seeing this, the slave device starts to work.
  - When it is finished it asserts SSYN’ (Slave SYNchronization).
  - Seeing this, the master reads the data.
  - When it is done, it negates MREQ’, RD’, the address lines, MSYN’ and SSYN’.



Synchronous - Disadvantages

- Although synchronous clocks are simple, there are some disadvantages
  - Everything done in multiples of clock, so something finishing in 3.1 cycles takes 4 cycles
  - With a mixture of fast and slow devices, we have to wait for the slowest device
    - Faster devices can’t run at their capacity, all devices are tied to a fixed clock rate
    - Consider memory device speed faster than 10ns, no speedup increase for 100Mhz clock
- One solution: Use asynchronous bus

Synchronous Timing Diagram
Read Operation Timing



Clock

Start

Indicates read/address lines valid, noticed by memory

Read

Indicates we want to read, not write

Address Lines

Address from memory we want

delay

Data Lines

Data from memory

Acknowledge

Indicates data lines valid

Bus Clocking

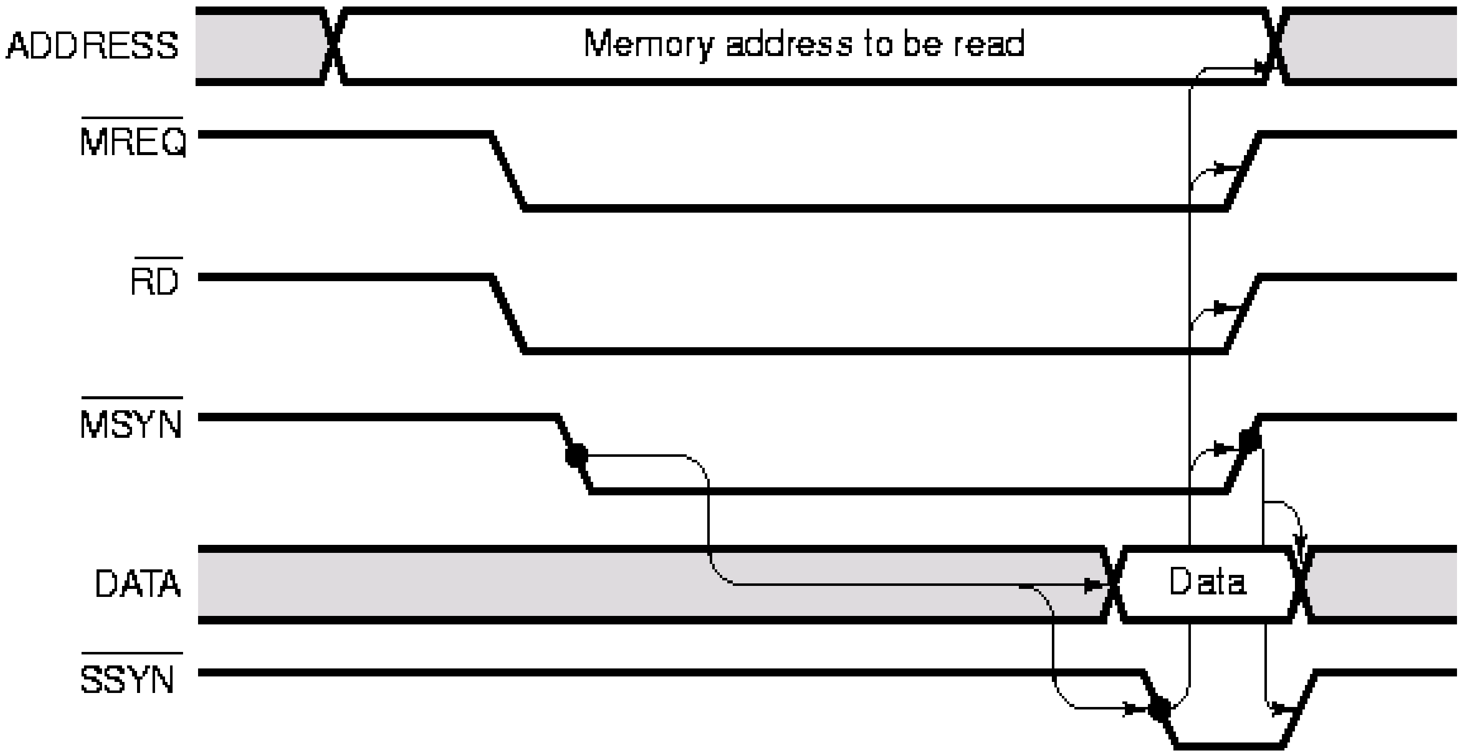
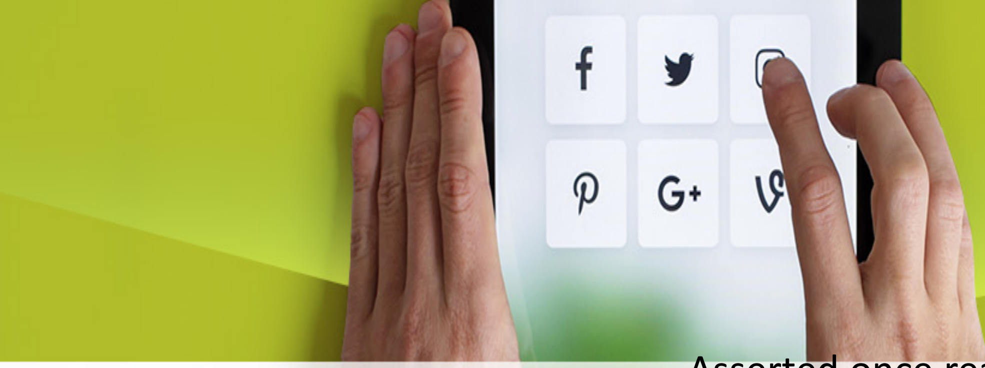


Figure 3-38. Operation of an asynchronous bus.

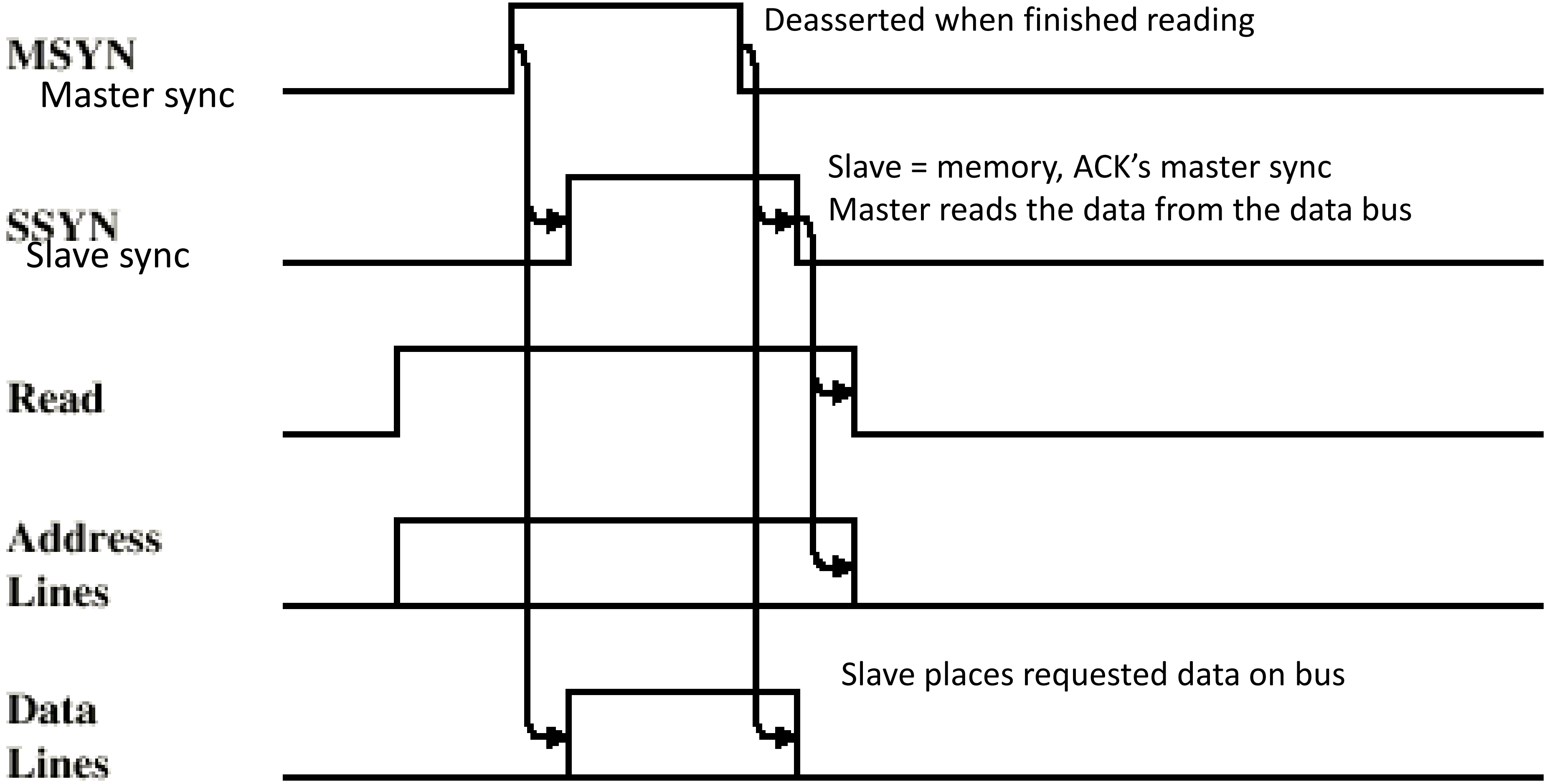
Asynchronous Bus

- No clock
- Occurrence of one event on the bus follows and depends on a previous event
- Requires tracking of state, hard to debug, but potential for higher performance

- Also used with networking
  – Problem with “drift” and loss of synchronization
  – Some use self-clocking codes, e.g. Ethernet



Asynchronous Timing Diagram



Asserted once read/address lines stabilize

Deasserted when finished reading

Master sync

Slave = memory, ACK's master sync

Master reads the data from the data bus

Slave sync

Slave places requested data on bus

Bus Arbitration

- I/O chips have to become bus master to read and write memory and to cause interrupts.
- If two or more devices want to become bus master at the same time, a **bus arbitration** mechanism is needed.
- Arbitration mechanisms can be centralized or decentralized. A simple form of centralized arbitration is shown on the next slide.
  - When the arbiter sees that one or more devices want to become master, it issues a grant by asserting the bus grant line.

Bus Arbitration

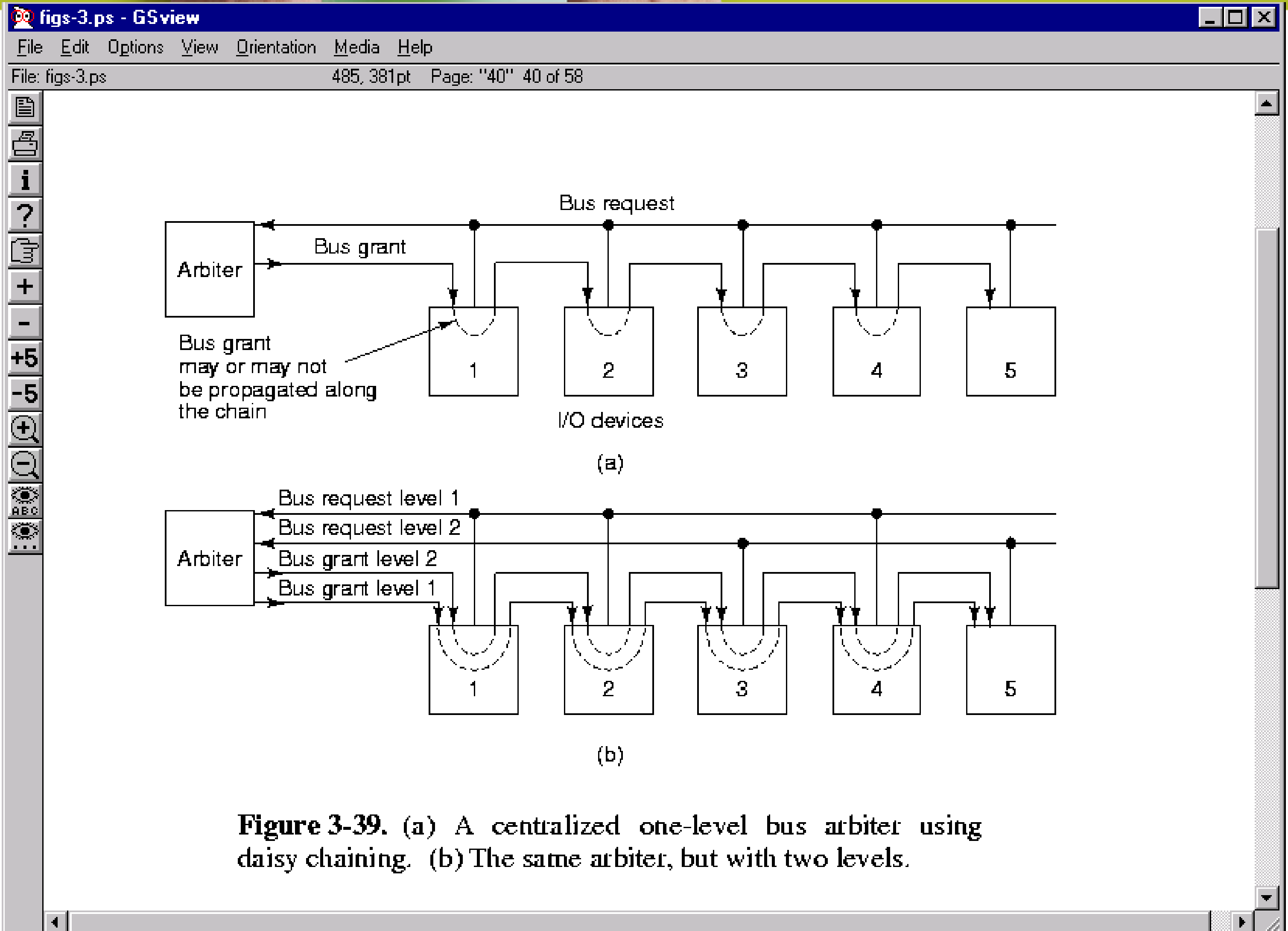


Figure 3-39. (a) A centralized one-level bus arbiter using daisy chaining. (b) The same arbiter, but with two levels.

Bus Arbitration

- In the first scheme shown, the closest device always wins.
- In the second scheme, there are multiple levels of priority. A device assert the line for its priority, and the arbiter grants the request by asserting the line with the highest priority.
- Since the CPU must compete for the device on most every cycle (i.e. it must read a word of memory) the memory is often put on a separate bus from the I/O devices so it doesn’t have to compete.

Bus Arbitration

- Decentralized bus arbitration is also possible.
  - A computer could have 16 prioritized bus request lines. When a device wants to use the bus, it assert its request line.
  - All devices monitor all request lines, so at the end of each bus cycle, each device knows whether it was the highest priority requester.
  - This method avoids the necessity of an arbiter, but requires more bus lines.
  - Another decentralized scheme equivalent to the daisy chain arbitration minus the arbiter is shown on the following slide.

Bus Arbitration

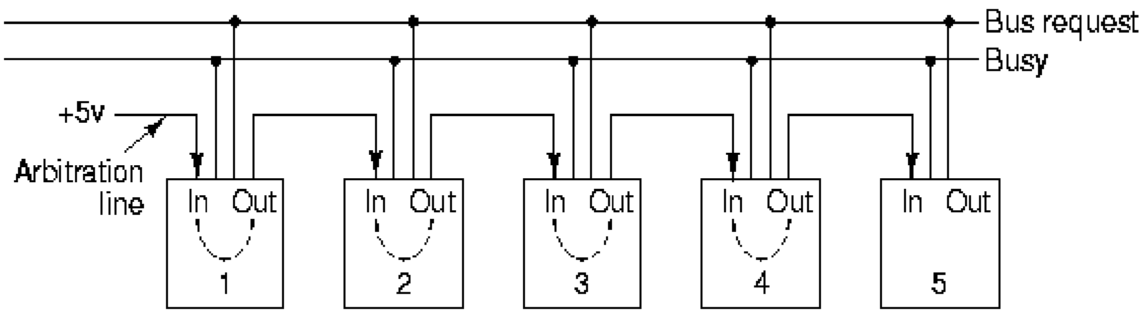


Figure 3-40. Decentralized bus arbitration.

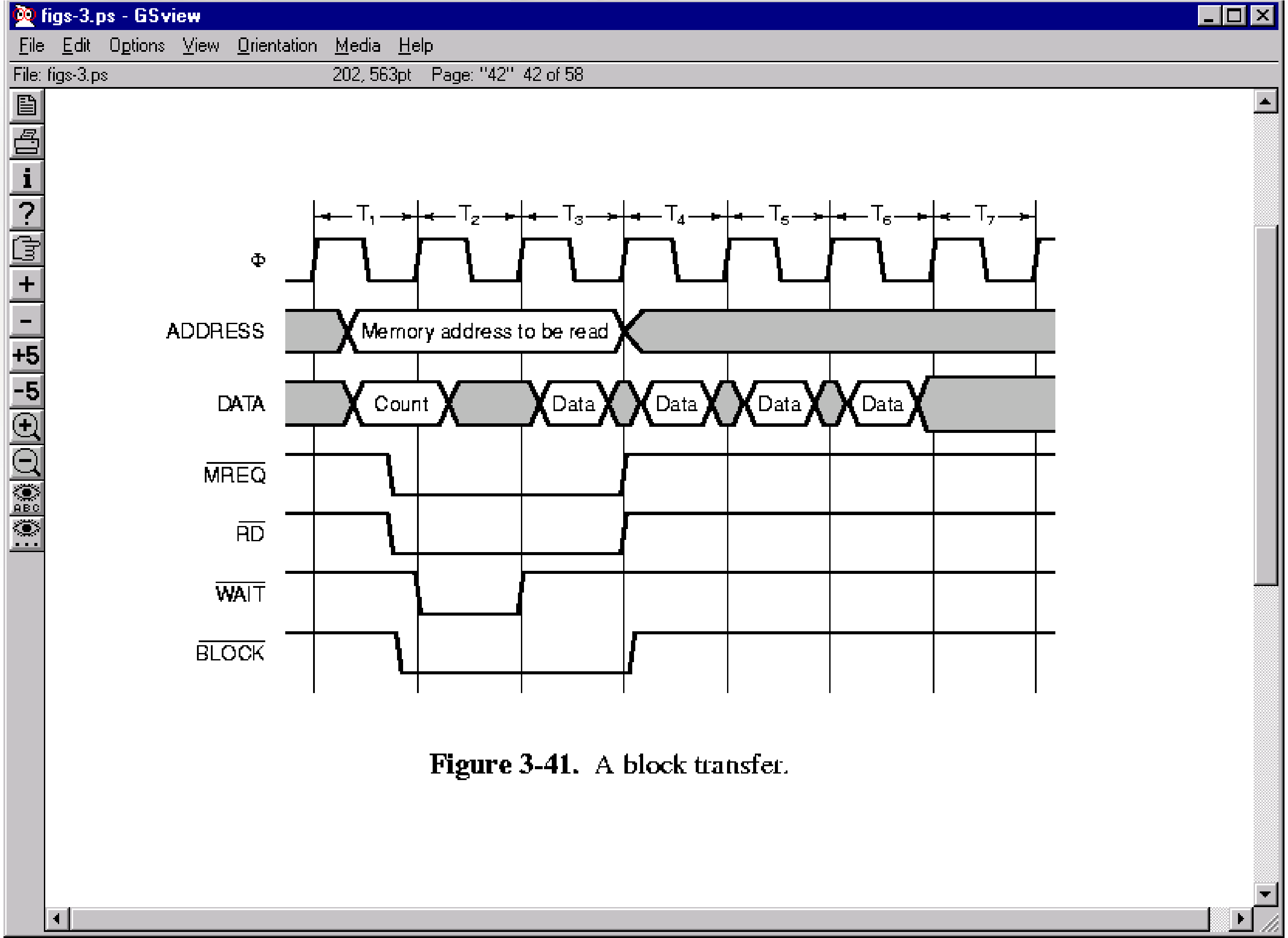
Bus Operations

– Up until now, we have only considered ordinary bus cycles, with a master reading from a slave or writing to one. In fact, several other kinds of bus cycles exist.

– Normally one word at a time is transferred. However, when caching is used it is often desirable to fetch an entire cache line at once.

- Block transfers can often be more efficient than successive individual transfers. The master puts the number of words to be transferred on the data lines during the first bus cycle.

Bus Operations



figs-3.ps - GSview

File: figs-3.ps 202, 563pt Page: "42" 42 of 58

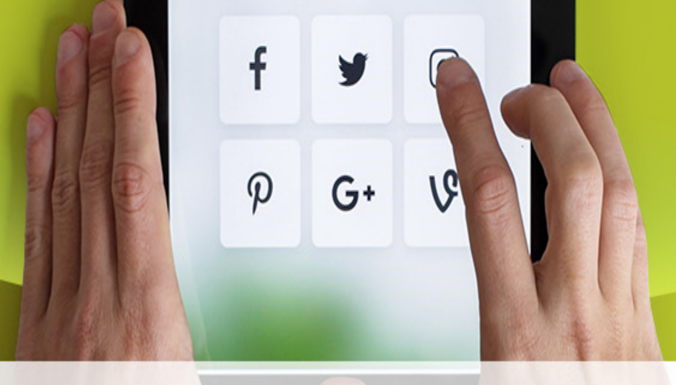
Figure 3-41. A block transfer.

Bus Operations

- Another important kind of bus cycle is for handling interrupts. When the CPU commands an I/O device to do something, it usually expects an interrupt when the work is done. The interrupt signaling requires the bus.
- Since multiple devices may want to cause an interrupt simultaneously, the same kind or arbitration problems we had with ordinary bus cycles are present.
  - The usual solution is to assign priorities and use a centralized arbiter.

Bus Operations

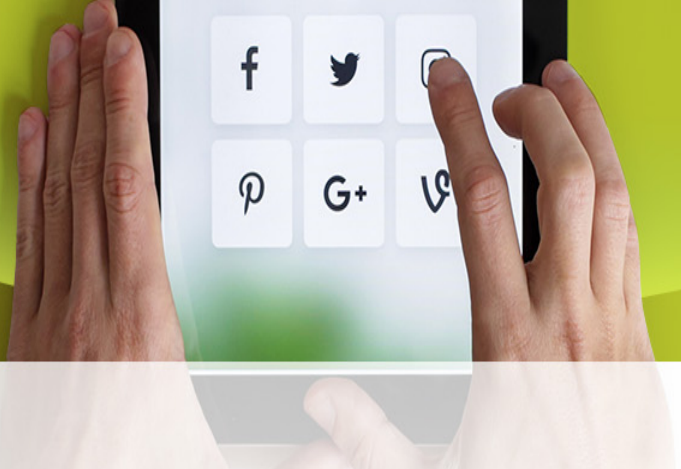
- Standard interrupt controller chips exist and are widely used.
- The IBM PC and all its successors use the Intel 8259A chip.
- Up to eight I/O controllers can be directly connected to the eight IR inputs to the 8259A. When one of these devices wants to cause an interrupt, it asserts its input line.
  - When one or more interrupts are asserted, the 8259A asserts INT which drives the interrupt pin on the CPU.



Bus Operations

- When the CPU is able to handle the interrupt, it sends back a pulse on INTA.
- At that point, the 8259A specifies which input caused the interrupt by outputting the input’s number on the data bus.
- The CPU uses that number to index into a table of pointers called interrupt vectors, to find the address of the procedure to run to service the interrupt.
    - Several 8259As can be cascaded to handle more than eight I/O devices.

Bus Operations



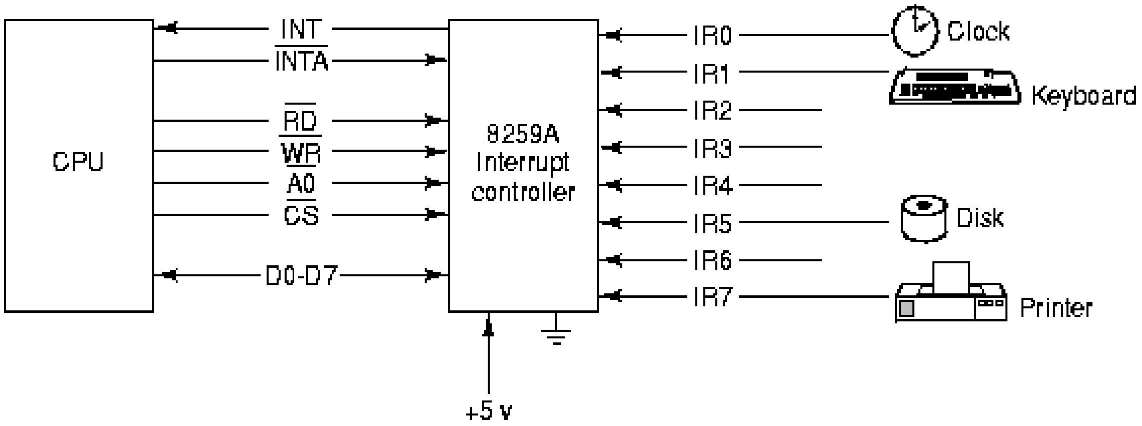
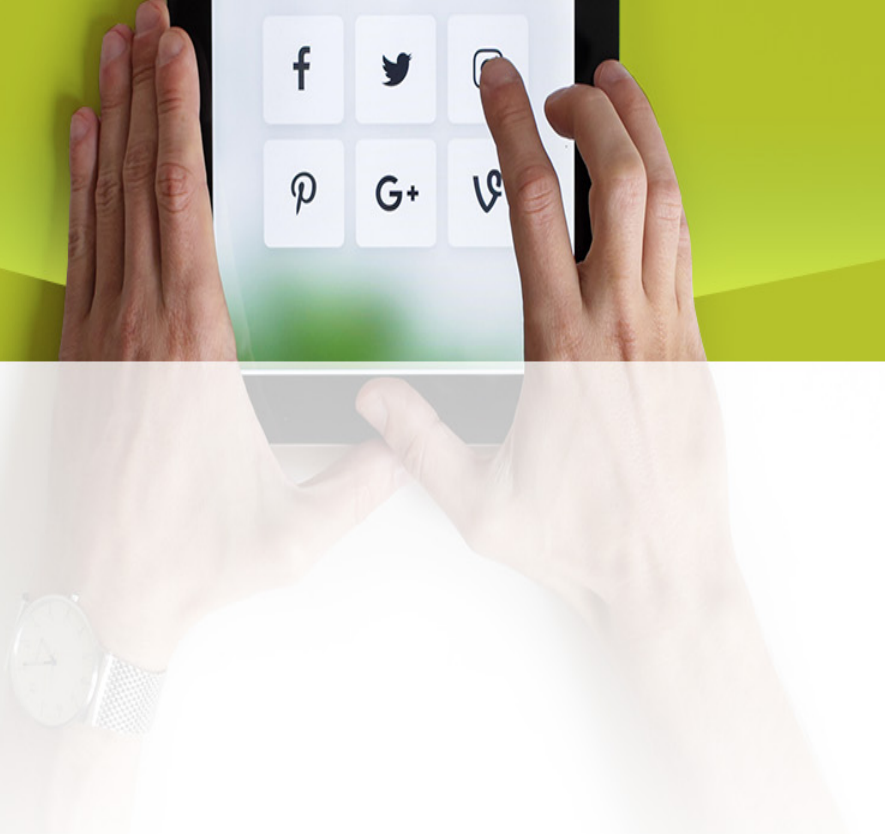
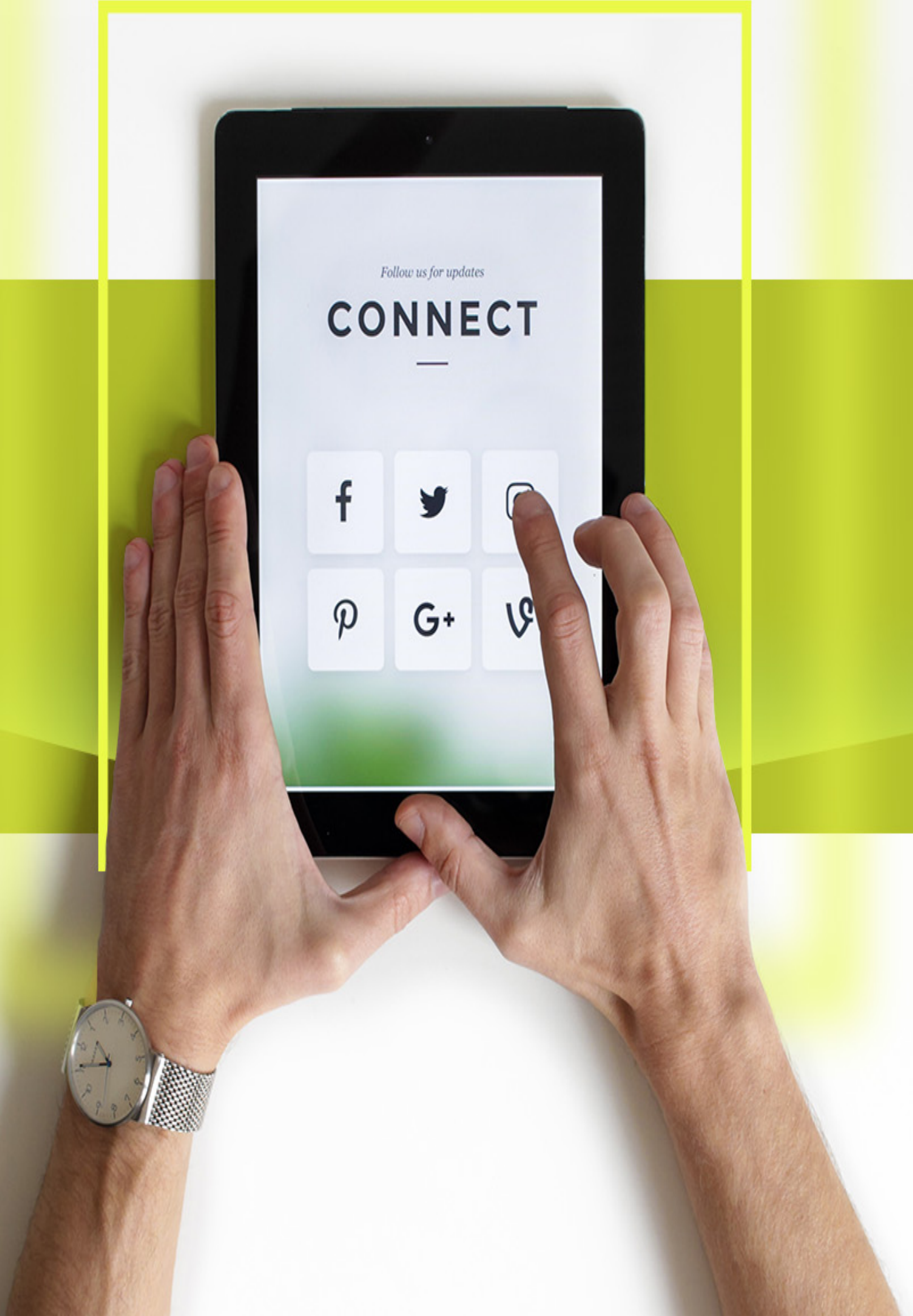


Figure 3-42. Use of the 8259A interrupt controller.



Lecture 2 Chapter 3

MEMORY



Memory

Types of Storage

- Secondary
  - Data that will eventually be used
  - Long-term
- Memory
  - Data that will be used in the near future
  - Temporary
  - Faster access than storage
- Registers
  - Data immediately related to the operation being executed
  - Faster access than memory

Measuring Storage Capacity

KB – kilobyte

- 1024 bytes
- Some diskettes
- Cache memory

MB – megabyte

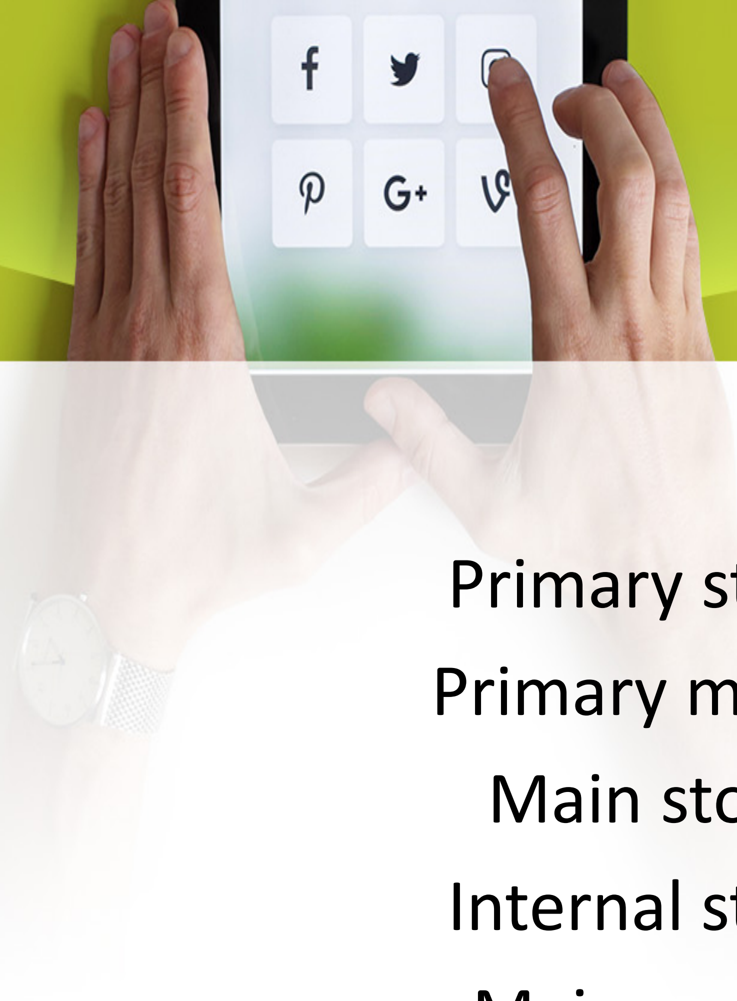
- Million bytes
- RAM

GB – gigabyte

- Billion bytes
- Hard disks
- CDs and DVDs

TB – terabytes

- Trillion bytes
- Large hard disks



Memory
Many Names

Primary storage

Primary memory

Main storage

Internal storage

Main memory

Main Types of Memory

RAM
Random Access Memory

ROM
Read Only Memory

RAM

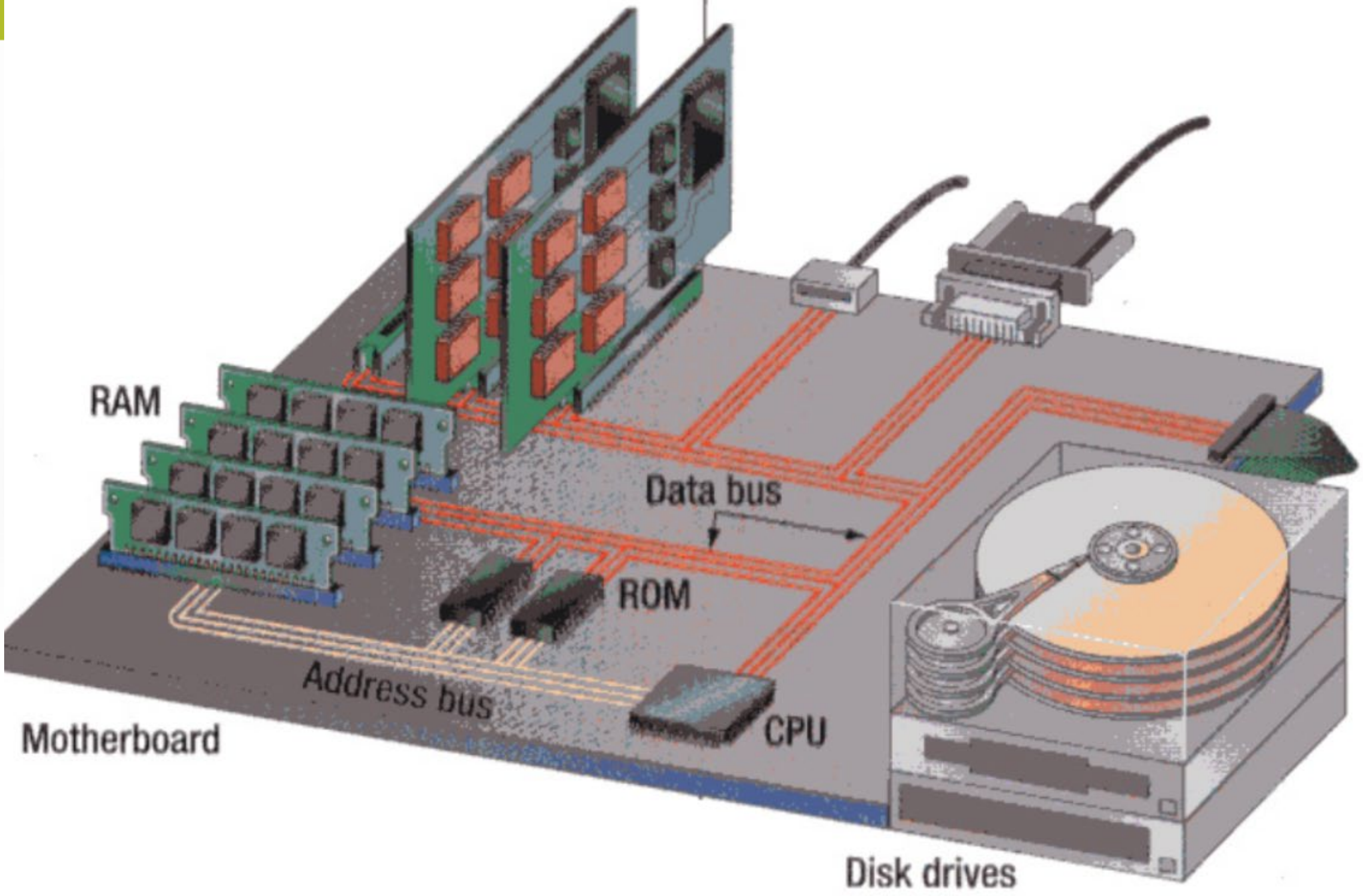
- Requires current to retain values
- Volatile
- Data and instructions can be read and modified
- Users typically refer to this type of memory

What’ s in RAM?

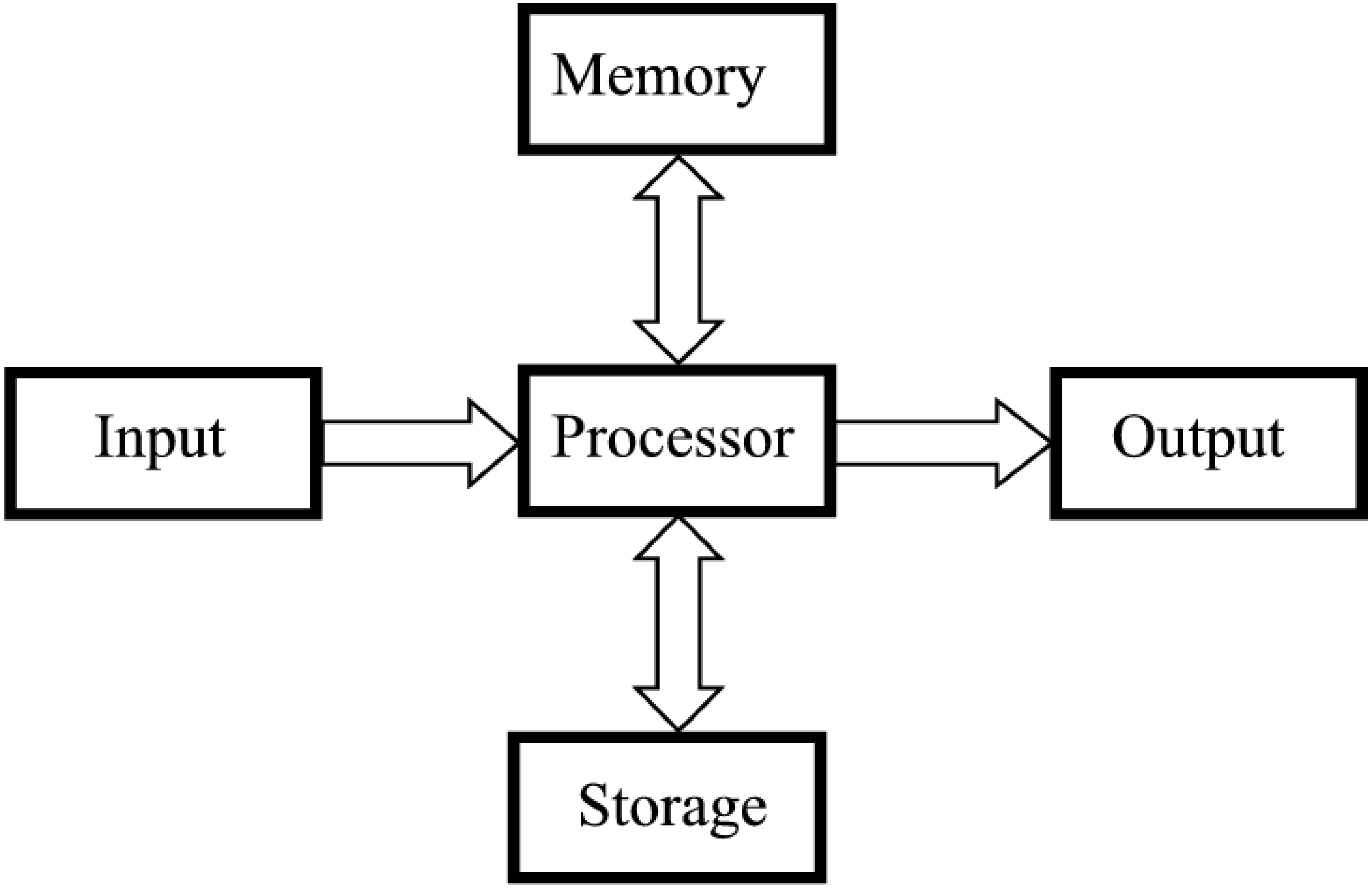
- Operating System
- Program currently running
- Data needed by the program
- Intermediate results waiting to be output

ROM

- Non-volatile
- Instructions for booting the computer
- Data and instructions can be read, but not modified
- Instructions are typically recorded at factory

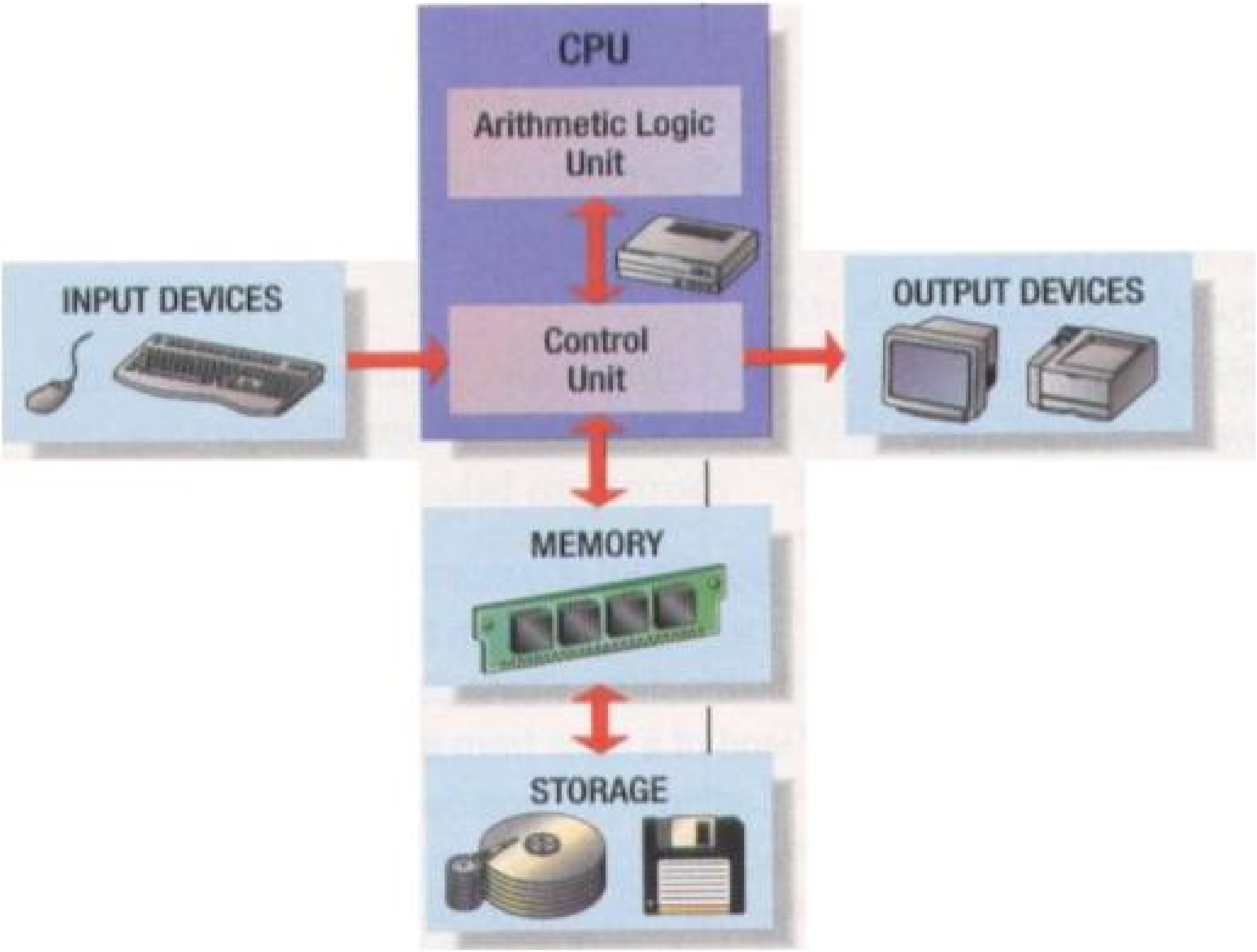

RAM
Data bus
ROM
Address bus
CPU
Motherboard
Disk drives

Memory



Introduction

- Memory Devices (RAM,ROM,PROM,EPROM)
- Storage Devices (Auxiliary Storage Devices-Magnetic Tape, Hard Disk, Floppy Disk .Optical Disks: CD-R Drive,CD-RW disks,DVD,Blue ray Discs)



CPU
Arithmetic Logic Unit
Control Unit
INPUT DEVICES
OUTPUT DEVICES
MEMORY
STORAGE

Characteristics of Storage Devices

- Speed
- Volatility
- Access method
- Portability
- Cost and capacity

Basic Units Of Measurement

- **Bit** on OR off

  _Bi_nary digi_t_

  Smallest unit of measurement

  Two possible values 0 1

- **Byte**

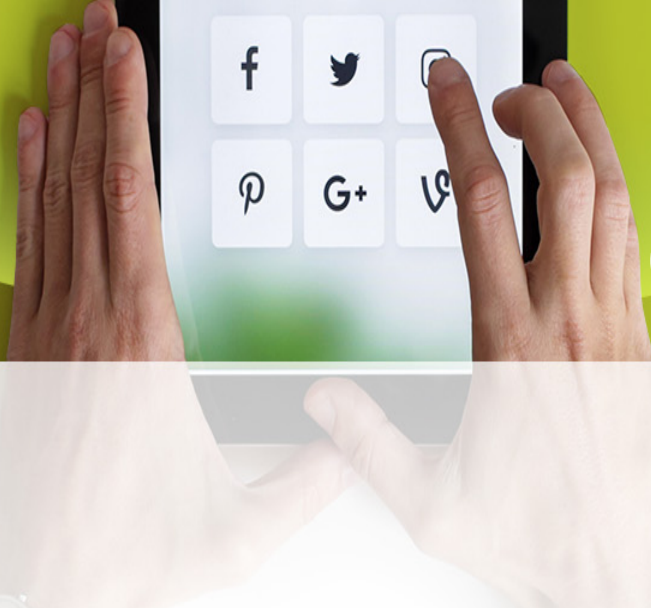
- **8 bits**

Small Units Of Measurement (Processor And Memory Speed)

Millisecond (ms) – a thousandth of a second (1/1,000 = 10^{-3})

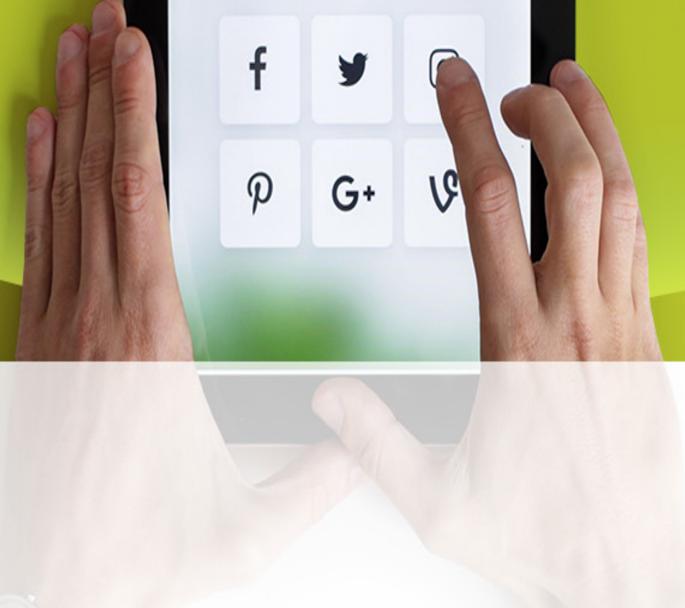
Microsecond (µs) - a millionth of a second (1/1,000,000 = 10^{-6})

Nanosecond (ns) – a billionth of a second (1/1,000,000,000 = 10^{-9})



Large Units Of Measurement (Memory, Storage)

- Note: powers of two are used because computer memory and storage are based on the basic unit (bit).
- Kilobyte (KB) – a thousand bytes (1,024 = 2^{10})
- Megabyte (MB) - a million (1,048,576 = 2^{20})



Large Units Of Measurement (Memory, Storage)

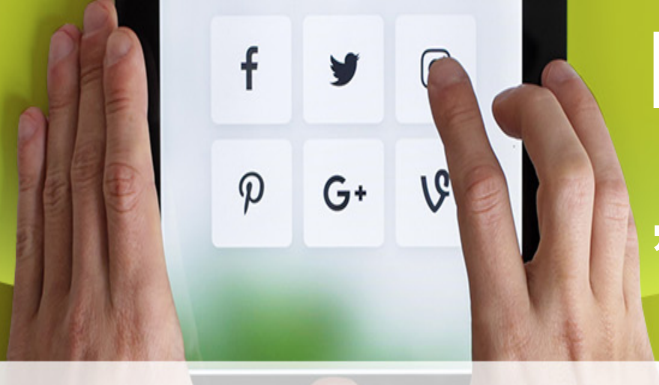
- Gigabyte (GB) – a billion (1,073,741,824 = 2^{30})
  - ~ A complete set of encyclopedias requires about 700 MB of storage
  - ~ 30 minutes of video (1/4 of the information stored on a typical DVD)

Large Units Of Measurement (Memory, Storage)

- Terabyte (TB) – a trillion (1,099,511,627,776 = 2^{40})
  - ~ 20 million four-drawer filing cabinets full of text
  - ~ 200 DVD's of information

RAM,ROM,PROM, EPROM

- Memory Devices
  - Memory: Is one or more sets of chips that store data/program instructions, either temporarily or permanently .
  - It is critical processing component in any computer
  - PCs use several different types



RAM,ROM,PROM ,EPROM

- Memory Devices
  - Two most important are
    - RAM(Random Access Memory)
    - ROM(Read-only Memory)
  - They work in different ways and perform distinct functions
  - CPU Registers
  - Cache Memory

RAM

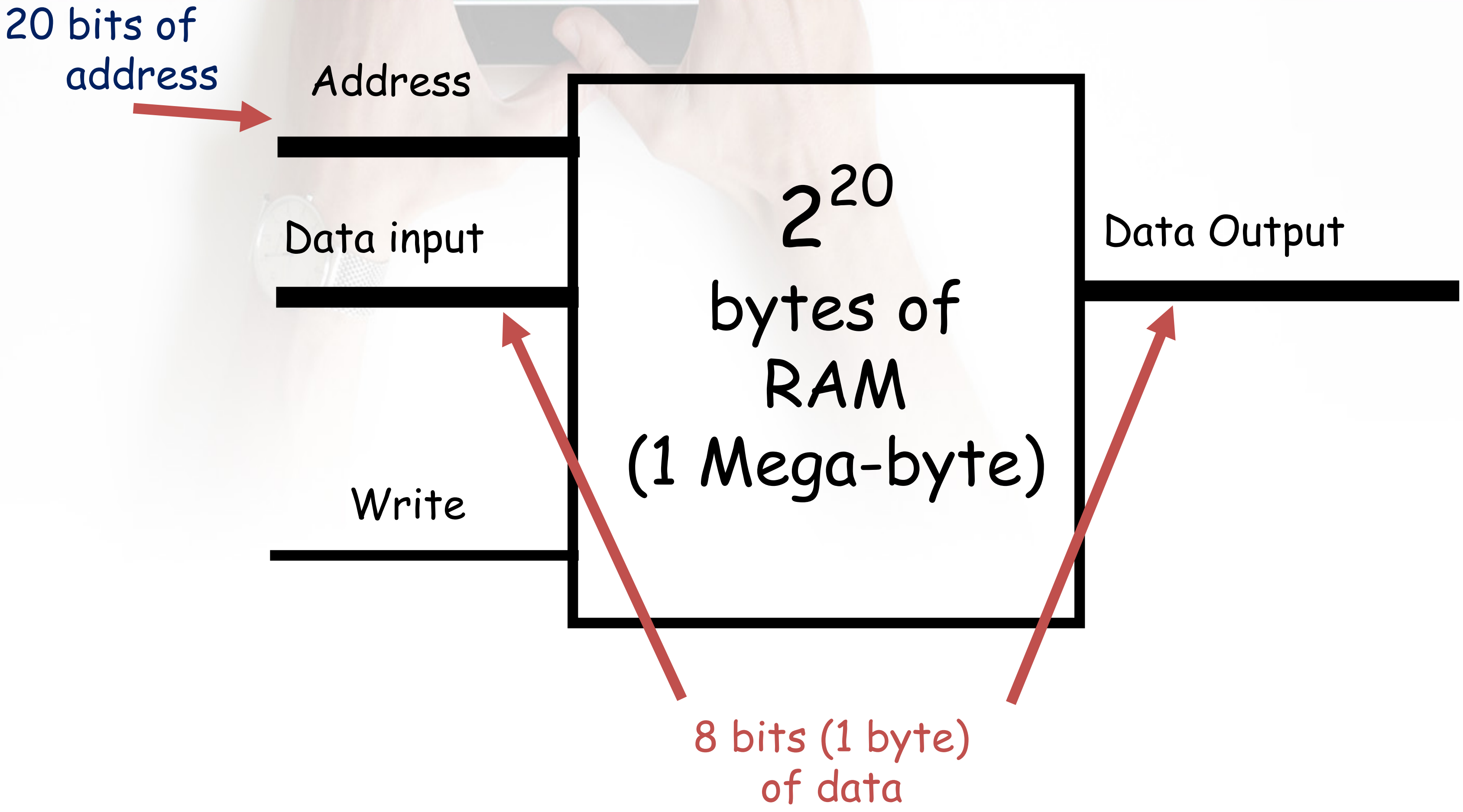
- RAM is packaged as a chip.
- Basic storage unit is a cell (one bit per cell).
- Multiple RAM chips form a memory.
- *R*andom *A*ccess *M*emory

  Volatile

  Used for temporary storage

  Typical ranges 256 MB - 4 GB
- Random Access means direct access to any part of memory

RAM



20 bits of address

Address

Data input

Write

2^{20} bytes of RAM (1 Mega-byte)

Data Output

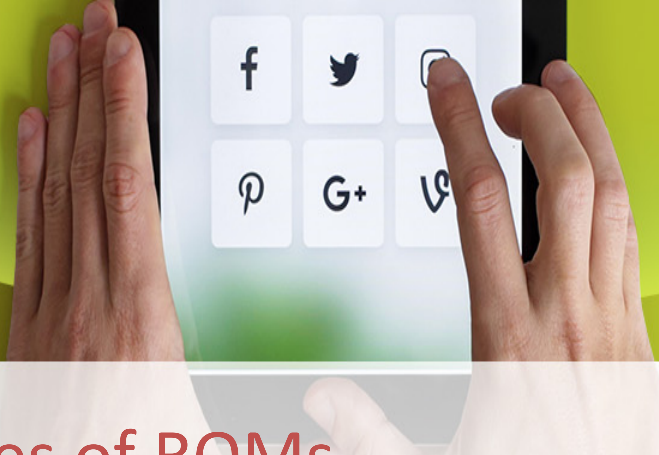
8 bits (1 byte) of data

RAM

- When you talk about the memory of a computer, most often you're talking about its RAM.
- If a program is stored in RAM, that means that a sequence of instructions are stored in consecutively addressed bytes in the RAM.
- Data values (variables) are stored anywhere in RAM, not necessarily sequentially
- Both instructions and data are accessed from RAM using addresses
- RAM is one (crucial) part of the computer's overall architecture

Nonvolatile Memories(ROM)

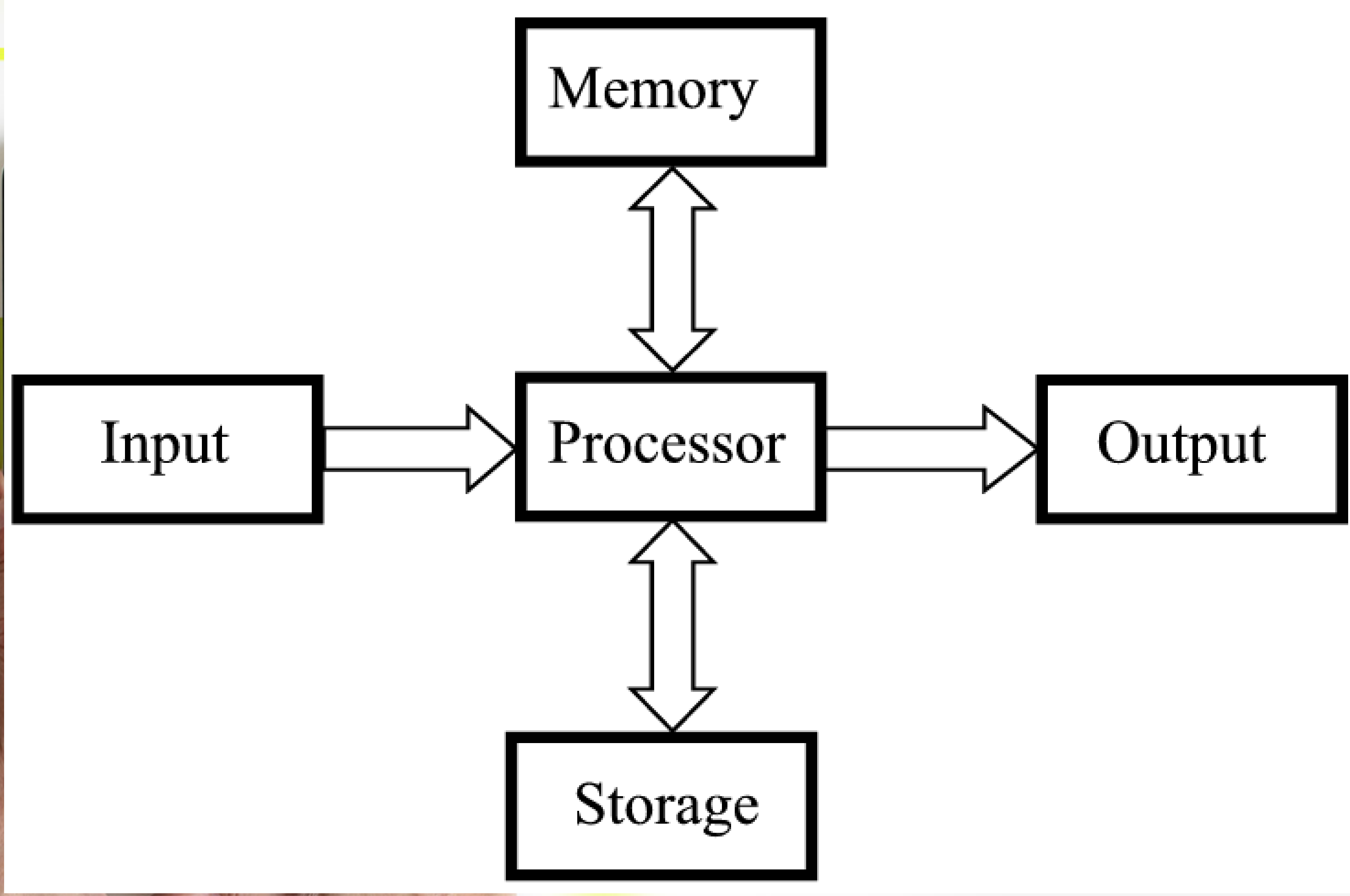
- DRAM and SRAM are volatile memories
  - Lose information if powered off.
- Nonvolatile memories retain value even if powered off.
  - Generic name is read-only memory (ROM).
  - Misleading because some ROMs can be read and modified.



Nonvolatile Memories(ROM)

- Types of ROMs
  - – Programmable ROM (PROM)
  - – Eraseable programmable ROM (EPROM)
  - – Electrically eraseable PROM (EEPROM)
  - – Flash memory (used in portable digital devices)
- Firmware (Program instruction used frequently)
  - – Program stored in a ROM
    - Boot time code, BIOS (basic input/output system)
    - graphics cards, disk controllers.

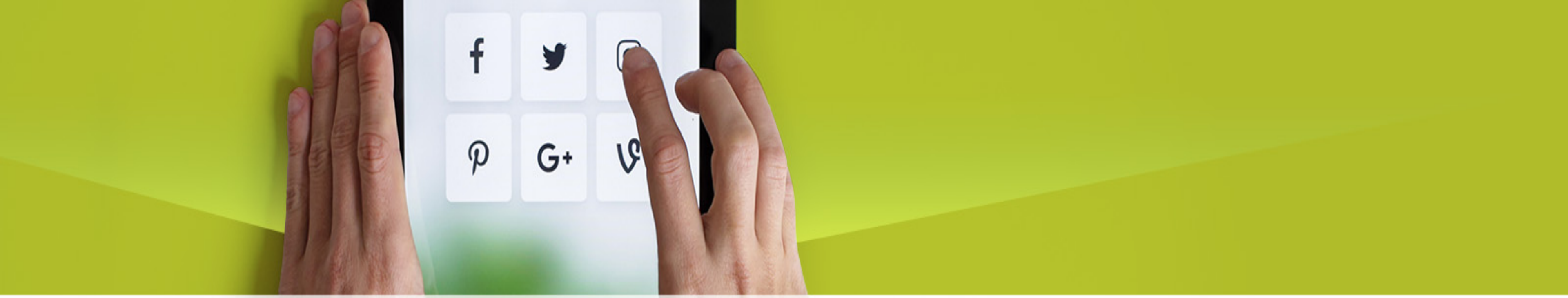
Memory



Storage Vs. Memory

Memory (e.g., RAM)

- Keep the information for a shorter period of time (usually volatile)
- Faster
- More expensive



3. Storage Vs. Memory

Storage (e.g., Hard disk)

- The information is retained longer (non-volatile)
- Slower
- Cheaper

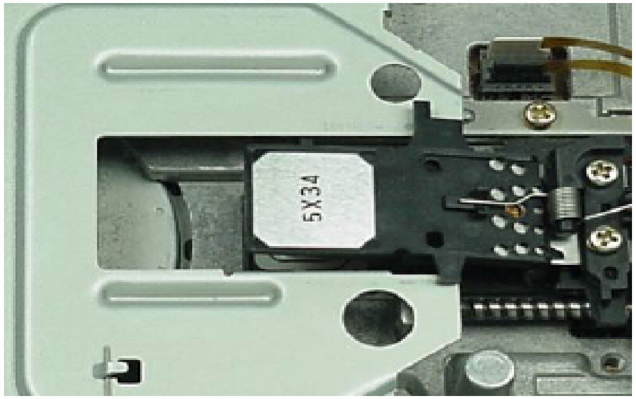
Categories Of Storage

- Magnetic
  - Floppy disks
  - Zip disks
  - Hard drives
- Optical
  - CD-ROM
  - DVD
- Solid state storage devices
  - USB Key (a very common form of solid state storage)

Magnetic Storage

- Exploits duality of magnetism and electricity
  - Converts electrical signals into magnetic charges
  - Captures magnetic charge on a storage medium
  - Later regenerates electrical current from stored magnetic charge
- Polarity of magnetic charge represents bit values zero and one

Magnetic Drives



Magnetic Disk

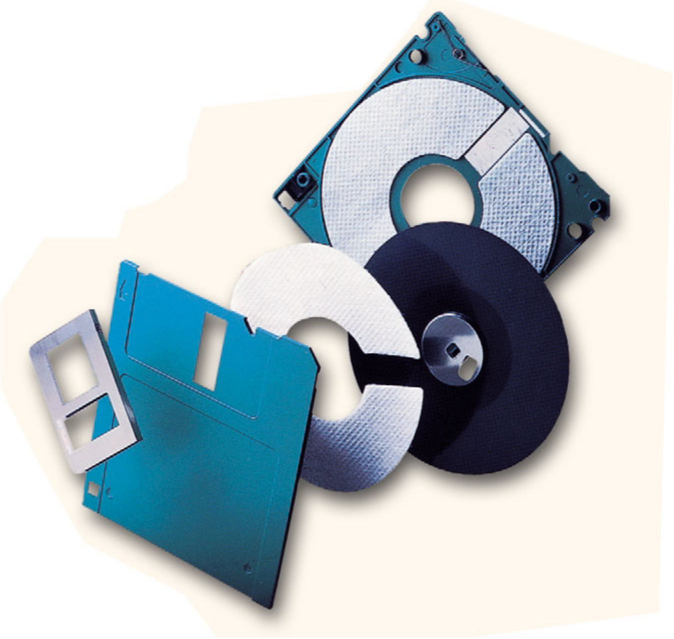
- Flat, circular platter with metallic coating that is rotated beneath read/write heads
- Random access device; read/write head can be moved to any location on the platter
- Hard disks and floppy disks
- Cost performance leader for general-purpose on-line secondary storage

1. Magnetic Drives: Storage Capacities

- Floppy disks
  - ~ 1 MB
- Hard drives
  - ~80 – 500 GB (TB is possible but very rare)

Floppy Disks

- A floppy disk is a portable, inexpensive storage medium that consists of a thin, circular, flexible plastic disk with a magnetic coating enclosed in a square-shaped plastic shell.

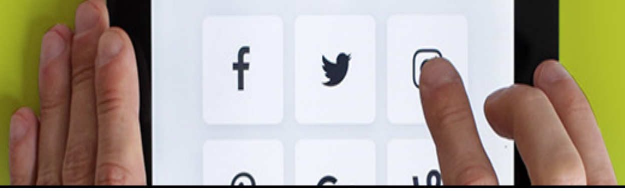


Structure Of Floppy Disks

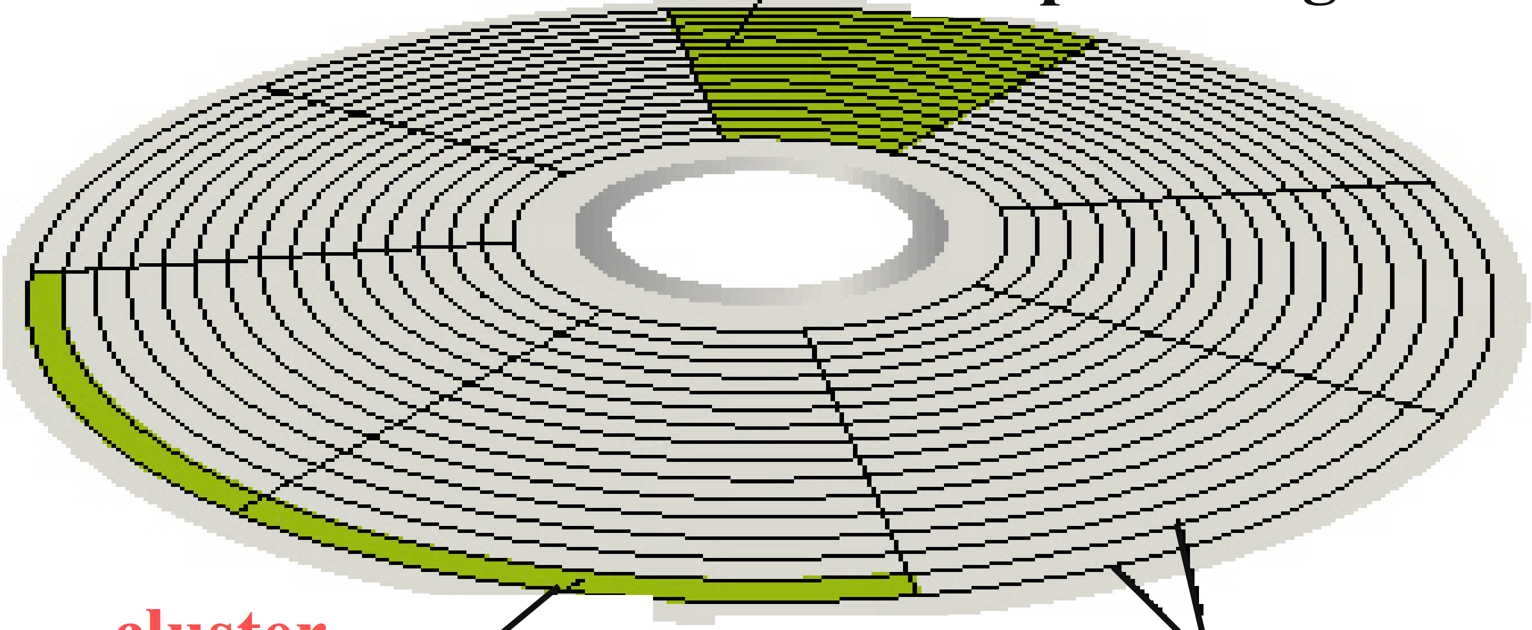
- Initially Floppy disks were 8-inches wide, they then shrank to 5.25 inches, and today the most widely used folly disks are 3.5 inches wide and can typically store 1.44 megabytes of data.
- A folly disk is a magnetic disk, which means that it used magnetic patterns to store data.
- Data in floppy disks can be read from and written to.
- **Formatting** is the process of preparing a disk for reading and writing.
- A track is a narrow recording band that forms a full circle on the surface of the disk.

Hard Disks

- Another form of auxiliary storage is a hard disk. A hard disk consists of one or more rigid metal plates coated with a metal oxide material that allows data to be magnetically recorded on the surface of the platters.
- The hard disk platters spin at a high rate of speed, typically 5400 to 7200 revolutions per minute (RPM).
- Storage capacities of hard disks for personal computers range from 10 GB to 120 GB (one billion bytes are called a gigabyte).



sectors
each track is divided into pie-shaped wedges

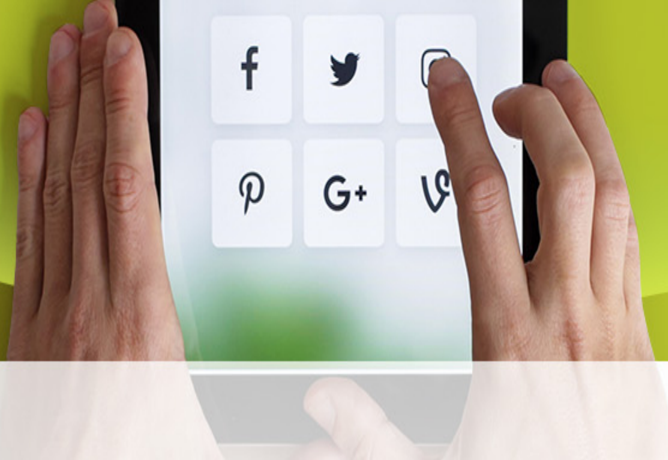


cluster
two or more sectors combined

tracks
data is recorded in concentric circular bands

Optical Mass Storage Devices

- Store bit values as variations in light reflection
- Higher areal density & longer data life than magnetic storage
- Standardized and relatively inexpensive
- Uses: read-only storage with low performance requirements, applications with high capacity requirements & where portability in a standardized format is needed



Optical Drives

- CD's (Compact Disk)

~ 700 MB storage

- –CD-ROM (read only)
- –CD-R: (**r**ecord) to a CD
- –CD-RW: can write and erase CD to reuse it (**r**e-**w**ritable)

- DVD(Digital Video Disk)

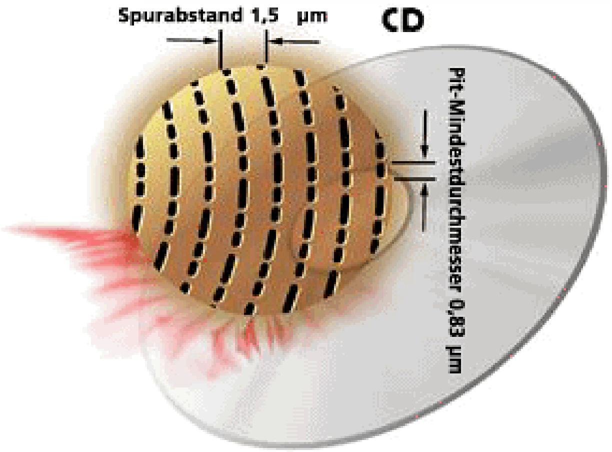
Compact Discs (CD)

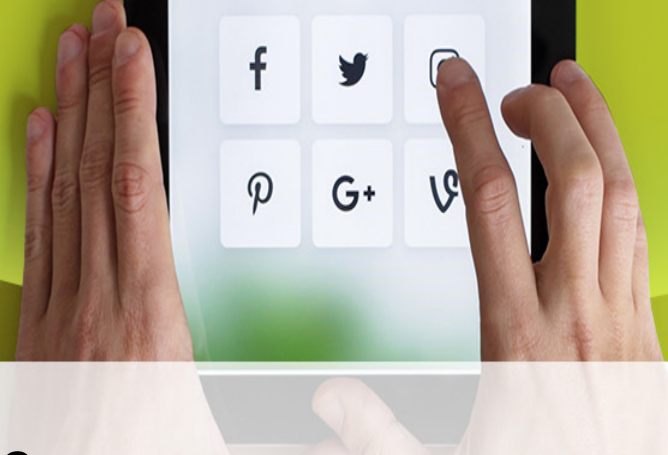
- A compact disk (CD), also called an optical disc, is a flat round, portable storage medium that is usually 4.75 inch in diameter.
- A CD-ROM (read only memory), is a compact disc that used the same laser technology as audio CDs for recording music. In addition it can contain other types of data such as text, graphics, and video.
- The capacity of a CD-ROM is 650 MB of data.

DVD (Digital Video Disk)

DVD-ROM

- Over 4 GB storage (varies with format)
- DVD- ROM (read only)
- Many recordable formats (e.g., DVD-R, DVD-RW; ..)
- Are more highly compact than a CD.
- Special laser is needed to read them





Blu-ray Technology

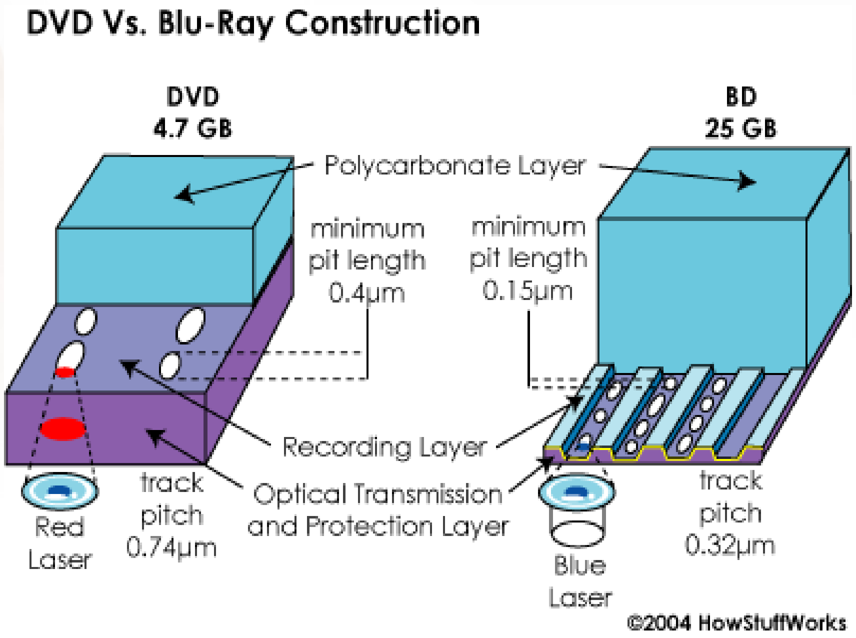
- Name

Derived from the blue-violet laser used to read and write data.

- Developed by the Blu-ray Disc Association with more than 180 members.
  - Dell
  - Sony
  - LG

Blu-ray Technology Cont.

- Data capacity
  - Because Blu-ray uses a blue laser(405 nanometers) instead of a red laser(650 nanometers) this allows the data tracks on the disc to be very compact.
  - This allows for more than twice as small pits as on a DVD.



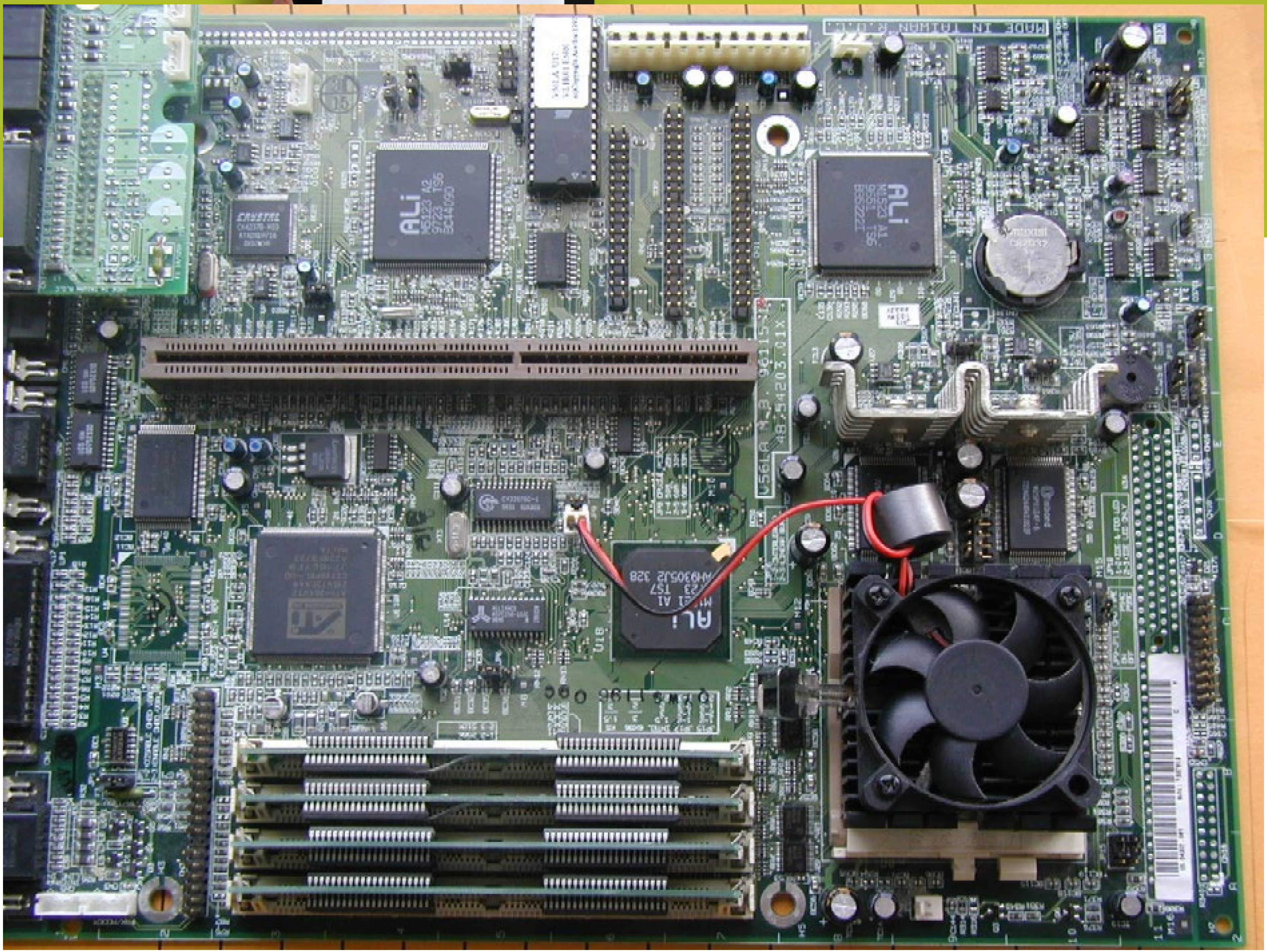
Blu-ray Technology Cont.

Formats

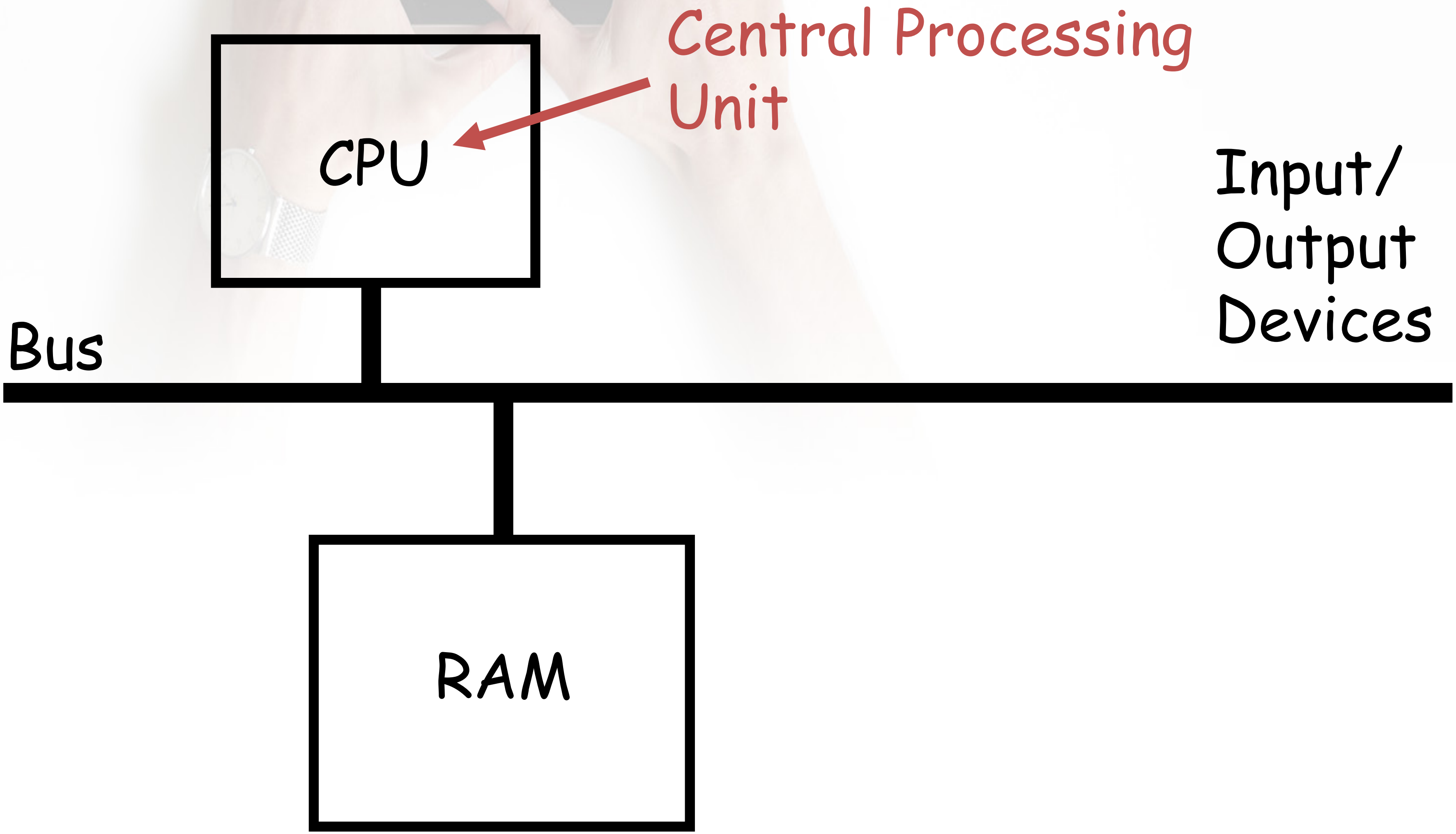
- BD-ROM (read-only) - for pre-recorded content
- BD-R (recordable) - for PC data storage
- BD-RW (rewritable) - for PC data storage
- BD-RE (rewritable) - for HDTV recording

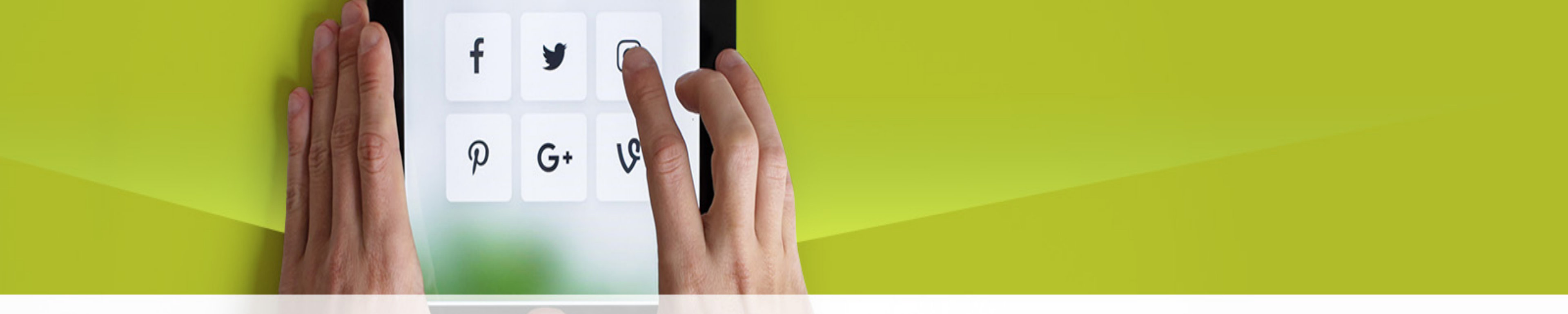
Lecture 2 Chapter 4

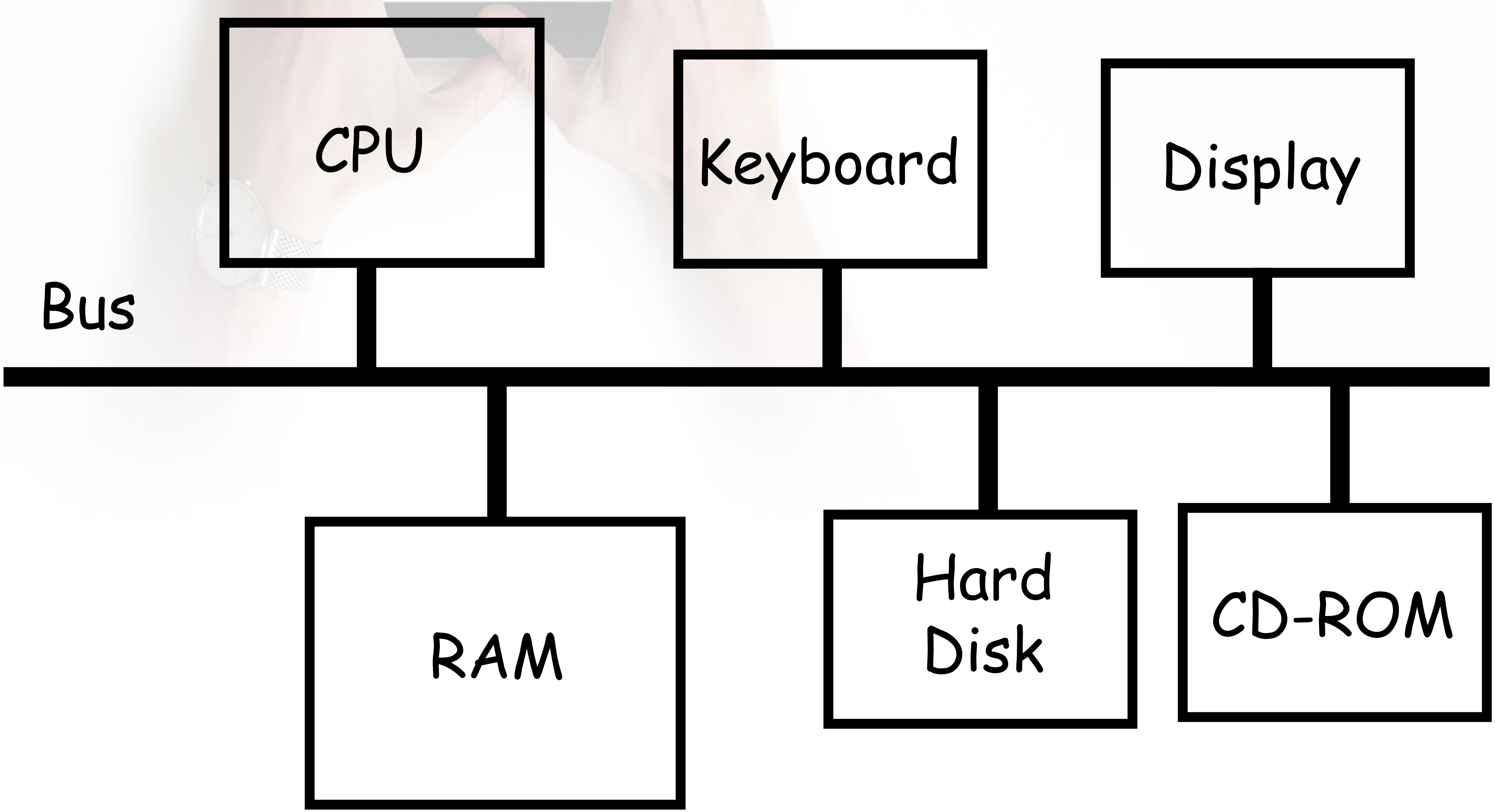
COMPUTER ARCHITECTURE


ALi
ALi
ALi
CRYSTAL

Computer Architecture







Bus

CPU

Keyboard

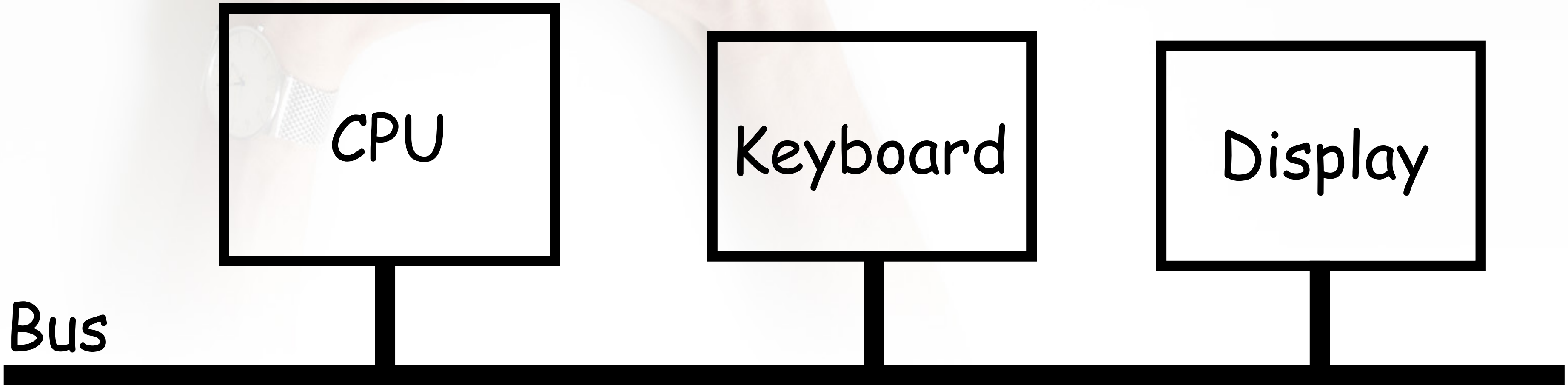
Display

RAM

Hard Disk

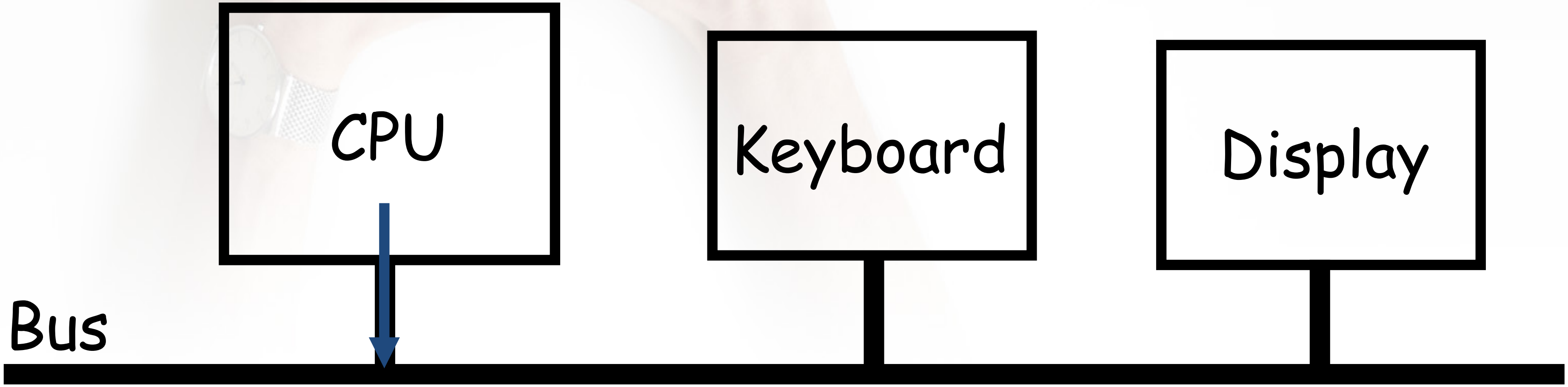
CD-ROM

The Bus



- Suppose CPU needs to check to see if the user typed anything.

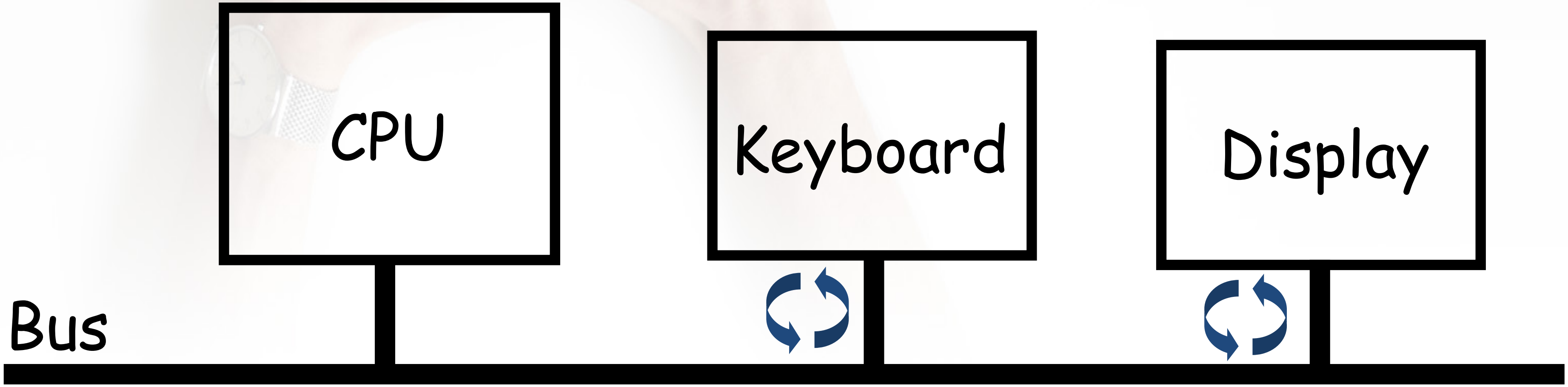
The Bus



“Keyboard, did the user type anything?”

- CPU puts “Keyboard, did the user type anything?” (represented in some way) on the Bus.

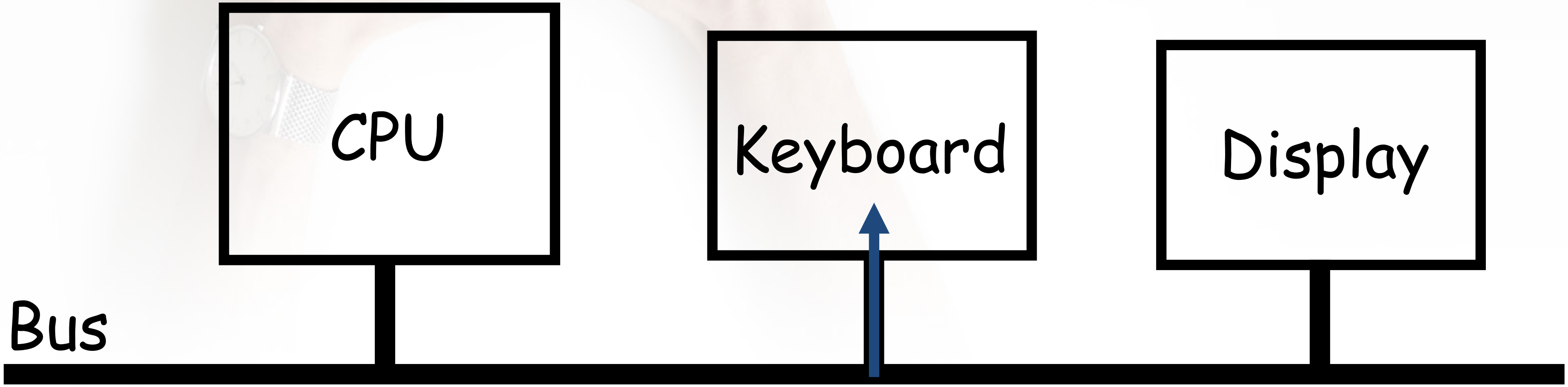
The Bus



"Keyboard, did the user type anything?"

- Each device (except CPU) is a State Machine that constantly checks to see what's on the Bus.

The Bus

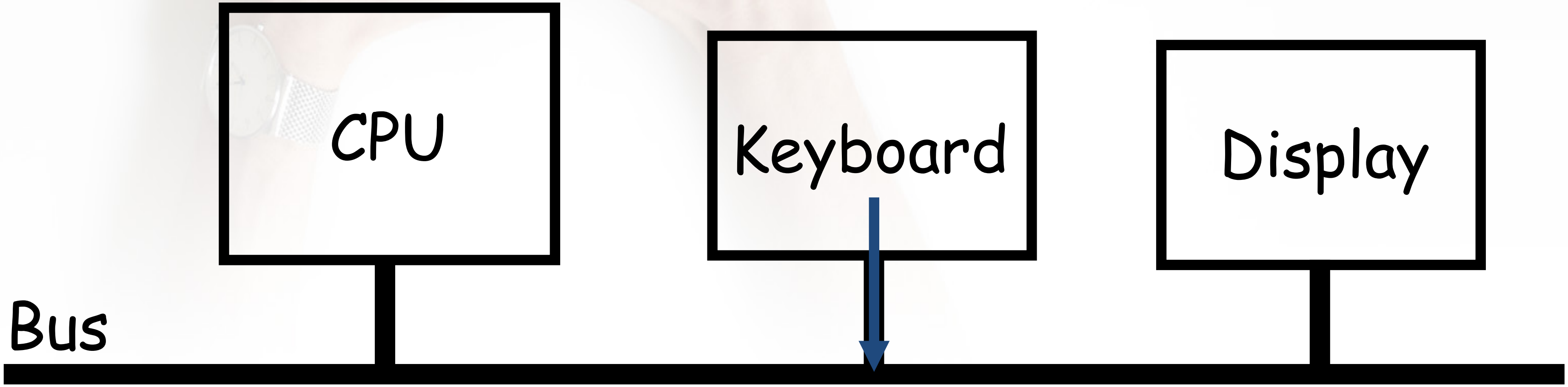


"Keyboard, did the user type anything?"

Keyboard notices that its name is on the Bus, and reads info. Other devices ignore the info.

The Bus

- Keyboard then writes "CPU: Yes, user typed 'a'." to the Bus.



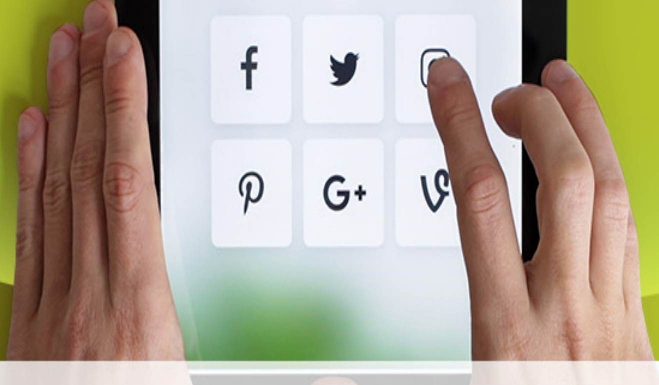
CPU

Keyboard

Display

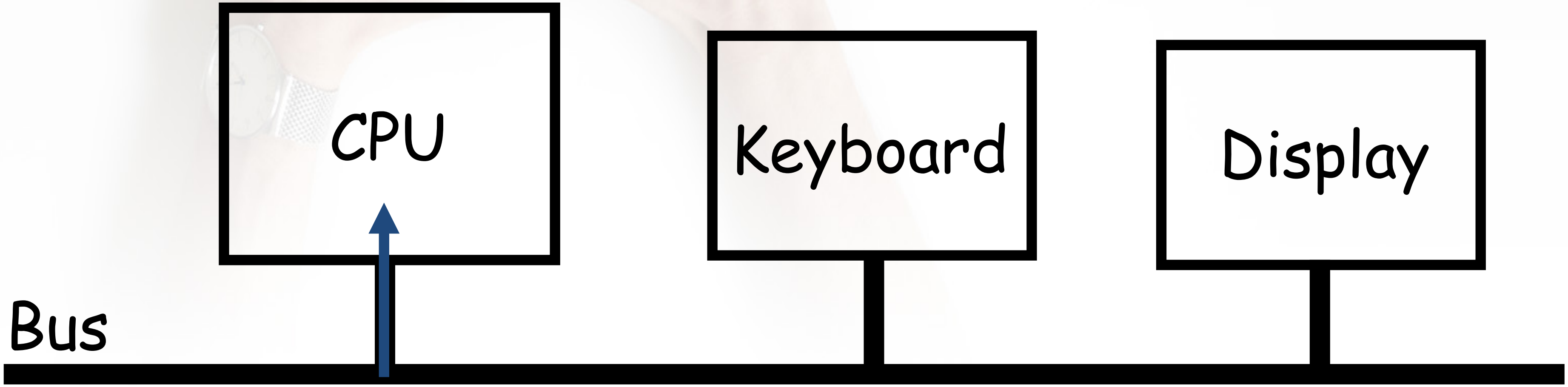
Bus

"CPU: Yes, user typed 'a'."

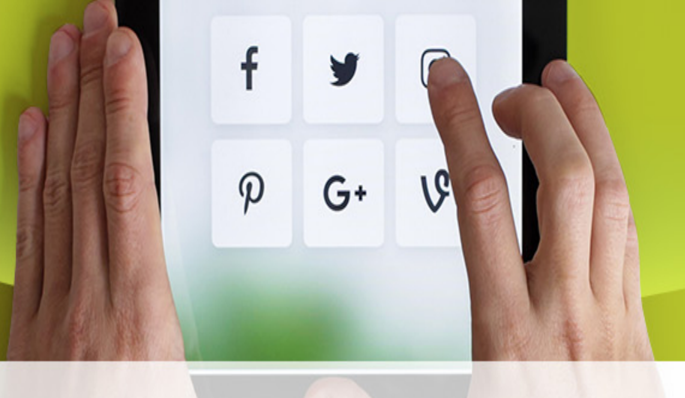


The Bus

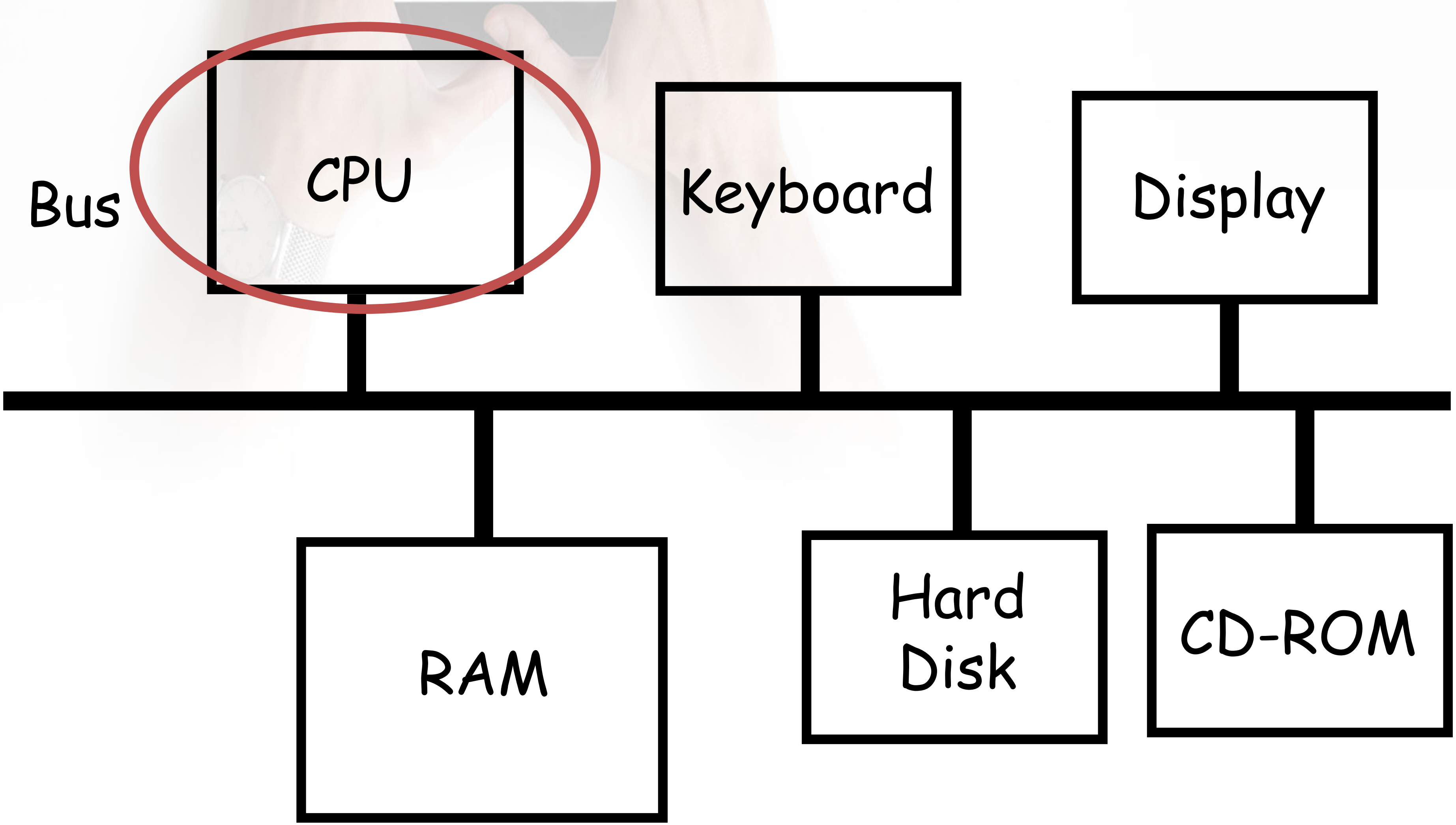
- At some point, CPU reads the Bus, and gets the Keyboard's response.



"CPU: Yes, user typed 'a'."



Computer Architecture



Bus

CPU

Keyboard

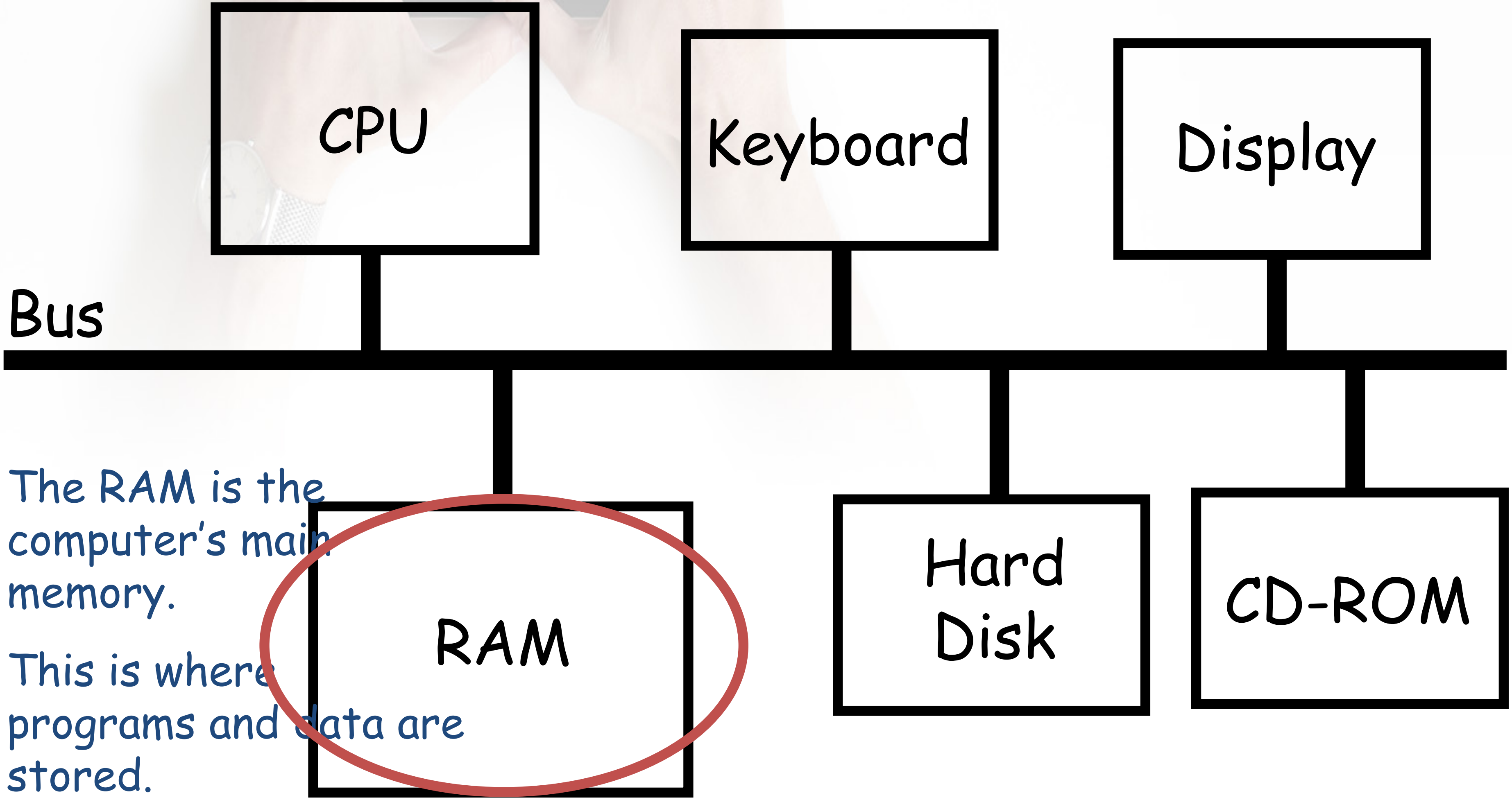
Display

RAM

Hard Disk

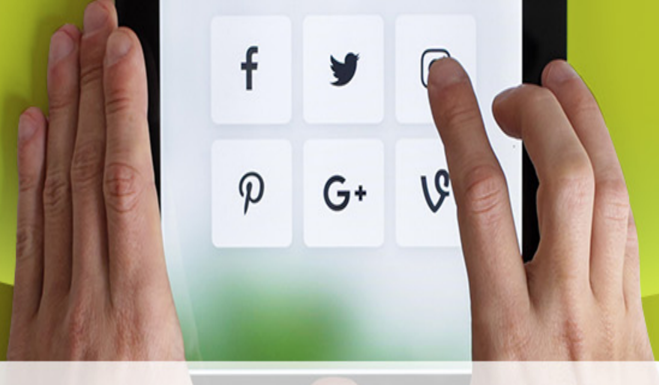
CD-ROM

Putting it all together

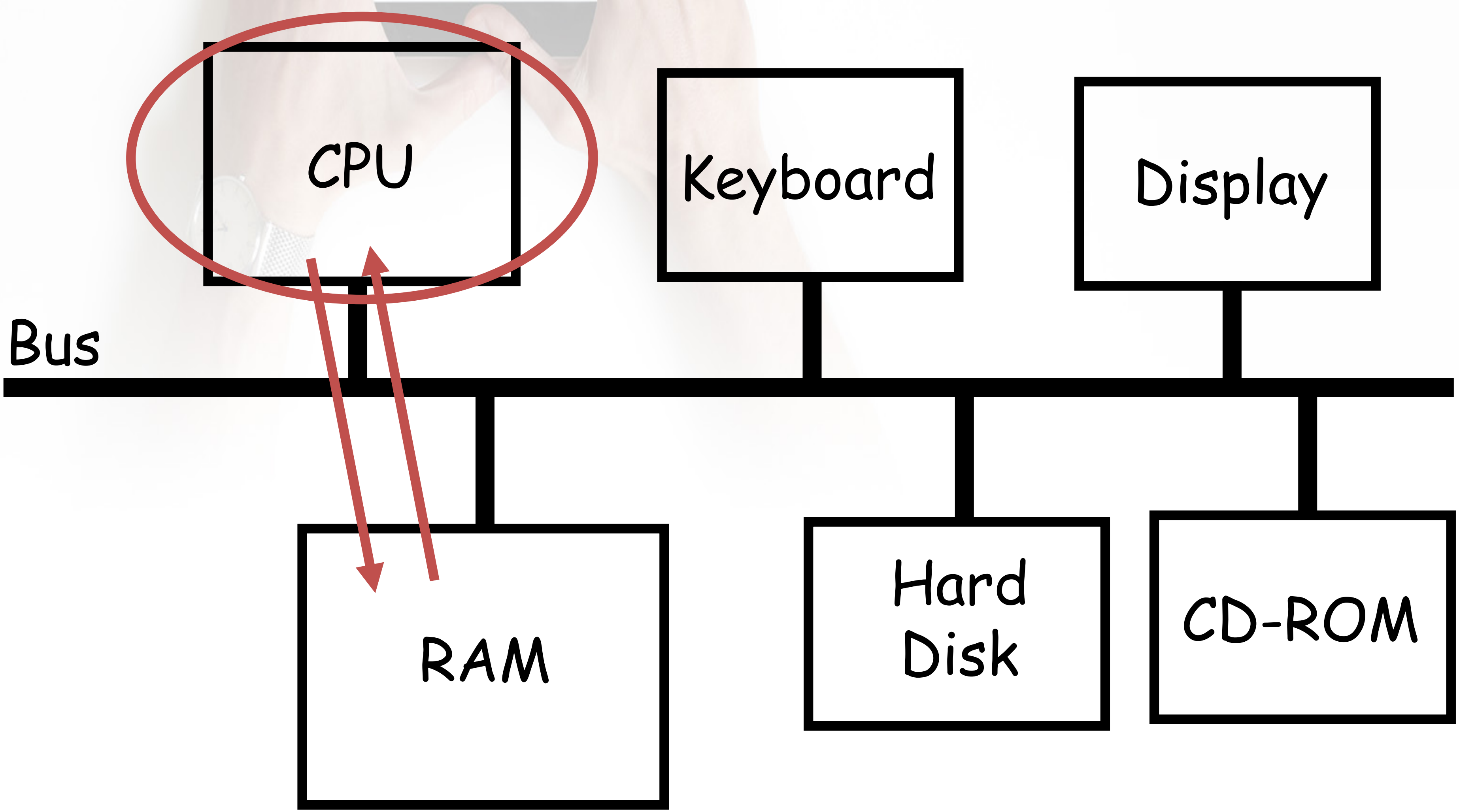


The RAM is the computer's main memory.

This is where programs and data are stored.



- The CPU goes in a never-ending cycle, reading instructions from RAM and executing them.



CPU

Keyboard

Display

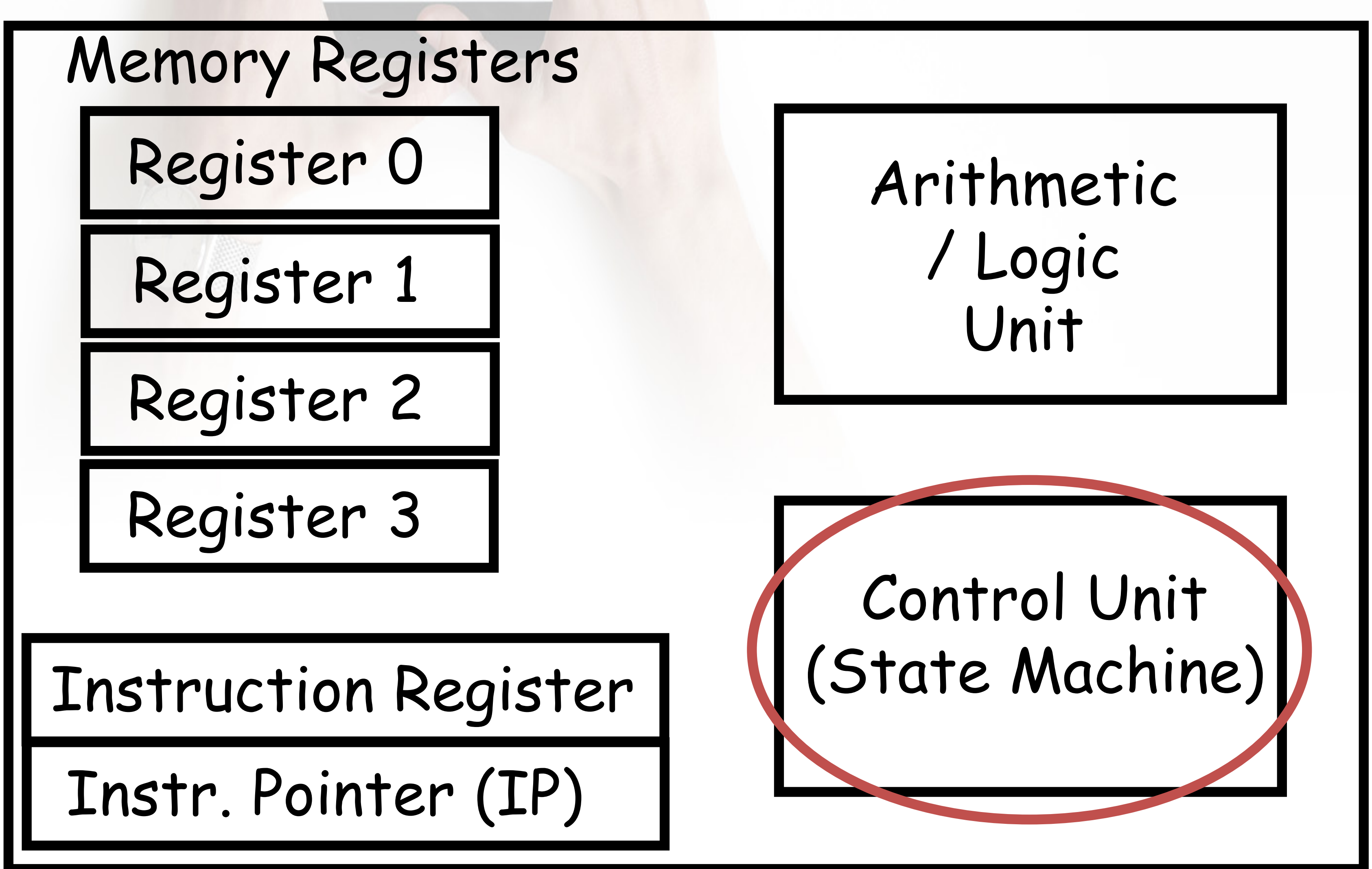
Bus

RAM

Hard Disk

CD-ROM

- This cycle is orchestrated by the Control Unit in the CPU.



Memory Registers

Register 0

Register 1

Register 2

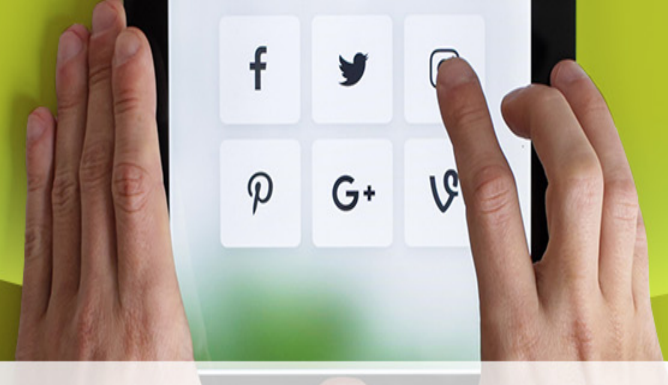
Register 3

Arithmetic / Logic Unit

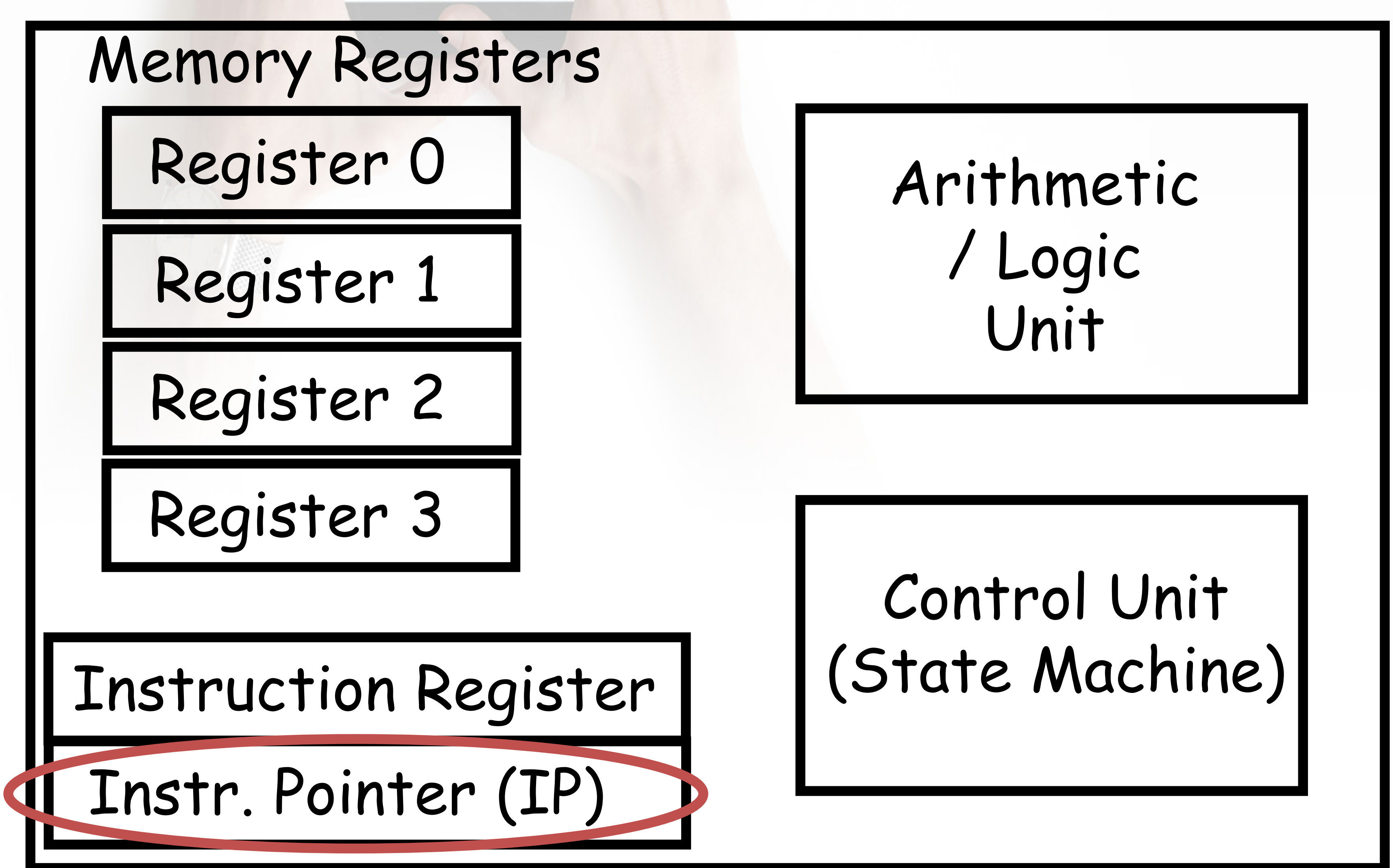
Control Unit (State Machine)

Instruction Register

Instr. Pointer (IP)



- It simply looks at where IP is pointing, reads the instruction there from RAM, and executes it.



Memory Registers

Register 0

Register 1

Register 2

Register 3

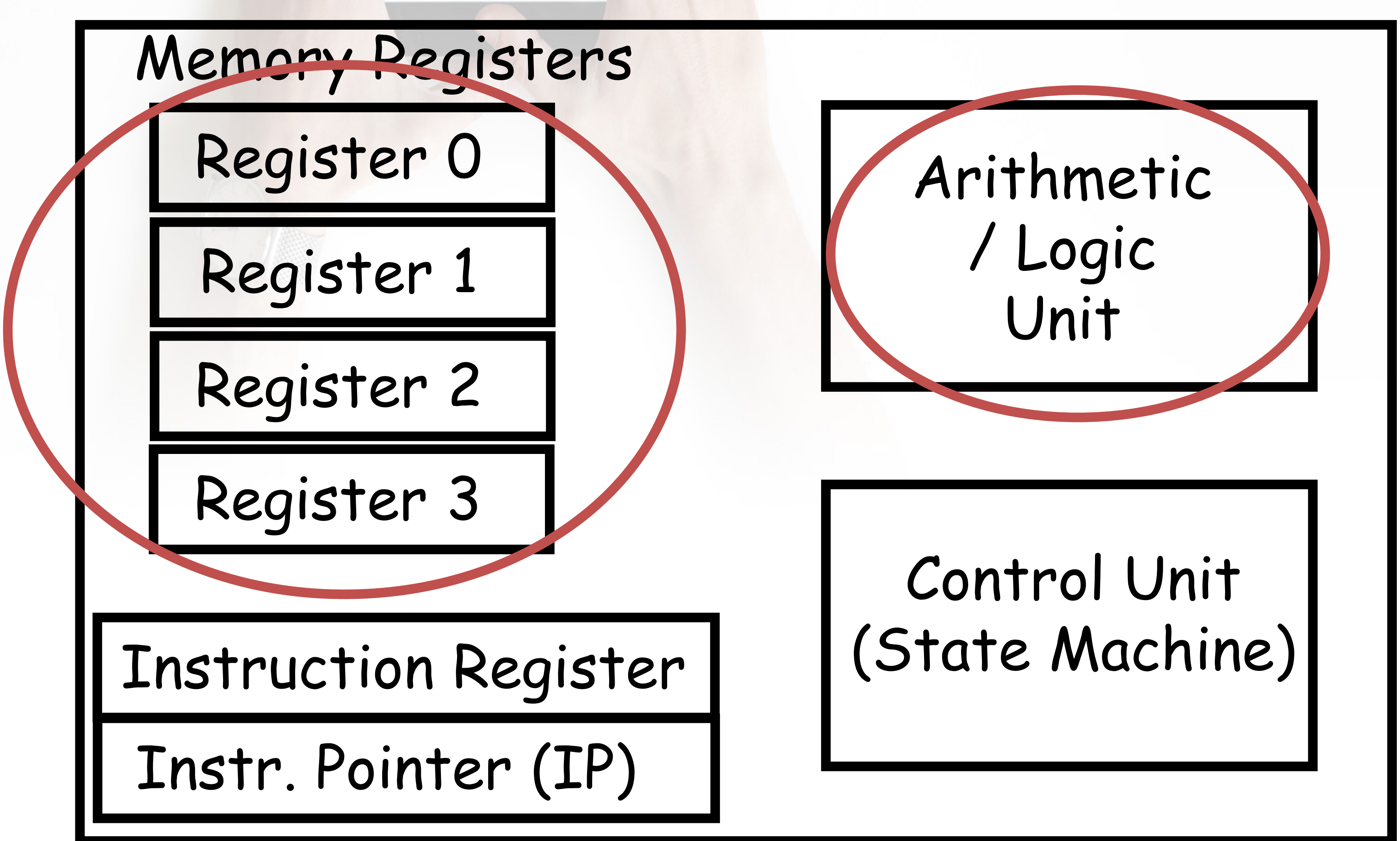
Instruction Register

Instr. Pointer (IP)

Arithmetic / Logic Unit

Control Unit (State Machine)

- To execute an instruction, the Control Unit uses the ALU plus Memory and/or the Registers.



Memory Registers

Register 0

Register 1

Register 2

Register 3

Instruction Register

Instr. Pointer (IP)

Arithmetic / Logic Unit

Control Unit (State Machine)

Programming

Where we are

- Examined the hardware for a computer
  - Truth tables
  - Logic gates
  - States and transitions in a state machine
  - The workings of a CPU and Memory
- Now, want to program the hardware

Programs and Instructions

- Programs are made up of instructions
- CPU executes one instruction every clock cycle
    - Modern CPUS do more, but we ignore that
- Specifying a program and its instructions:
    - Lowest level: Machine language
    - Intermediate level: Assembly language
    - Typically today: High-level programming language

Specifying a Program and its Instructions

- High-level programs: each statement translates to many instructions
  - E.g. c ← a + b to:

Load a into r1
Load b into r3
r2 ← r1 + r3
Store r2 into c

- Assembly language: specify each machine instruction, using mnemonic form
  - E.g. Load r1, *A*
- Machine language: specify each machine instruction, using bit patterns
  - E.g. 1101101000001110011

Machine/Assembly Language

- We have a machine that can execute instructions
- Basic Questions:
  - What instructions?
  - How are these instructions represented to the computer hardware?

Complex vs Simple Instructions

- Computers used to have very complicated instruction sets – this was known as:
  - CISC = Complex Instruction Set Computer
  - Almost all computers 20 years ago were CISC.
- 80s introduced RISC:
  - RISC = Reduced Instruction Set Computer

Complex vs Simple Instructions

- RISC = Reduced Instruction Set Computer
  - Fewer, Less powerful basic instructions
  - But Simpler, Faster, Easier to design CPU's
  - Can make "powerful" instructions by combining several wimpy ones

- Shown to deliver better performance than Complex Instruction Set Computer (CISC) for several types of applications.

Complex vs Simple Instructions

- Nevertheless, Pentium is actually CISC !
- Why?

Complex vs Simple Instructions

- Nevertheless, Pentium is actually CISC !
- Why: Compatibility with older software
- Newer application types (media processing etc) perform better with specialized instructions
- The world has become too complex to talk about RISC versus CISC

Typical Assembly Instructions

- Some common assembly instructions include:
  - 1) "Load" – Load a value from RAM into one of the registers
  - 2) "Load Direct" – Put a fixed value in one of the registers (as specified)
  - 3) "Store" - Store the value in a specified register to the RAM
  - 4) "Add" - Add the contents of two registers and put the result in a third register

Typical Assembly Instructions

- Some common instructions include:
  - 5) “Compare” - If the value in a specified register is larger than the value in a second register, put a “0” in Register r0
  - 6) “Jump” - If the value in Register r0 is “0”, change Instruction Pointer to the value in a given register
  - 7) “Branch” - If the value in a specified register is larger than that in another register, change IP to a specified value

Machine Languages

- Different types of CPU's understand different instructions
  - Pentium family / Celeron / Xeon / AMD K6 / Cyrix … (Intel x86 family)
  - PowerPC (Mac)
  - DragonBall (Palm Pilot)
  - StrongARM/MIPS (WinCE)
  - Many Others (specialized or general-purpose)
- They represent instructions differently in their assembly/machine languages (even common ones)
- Let's look instructions for a simple example CPU