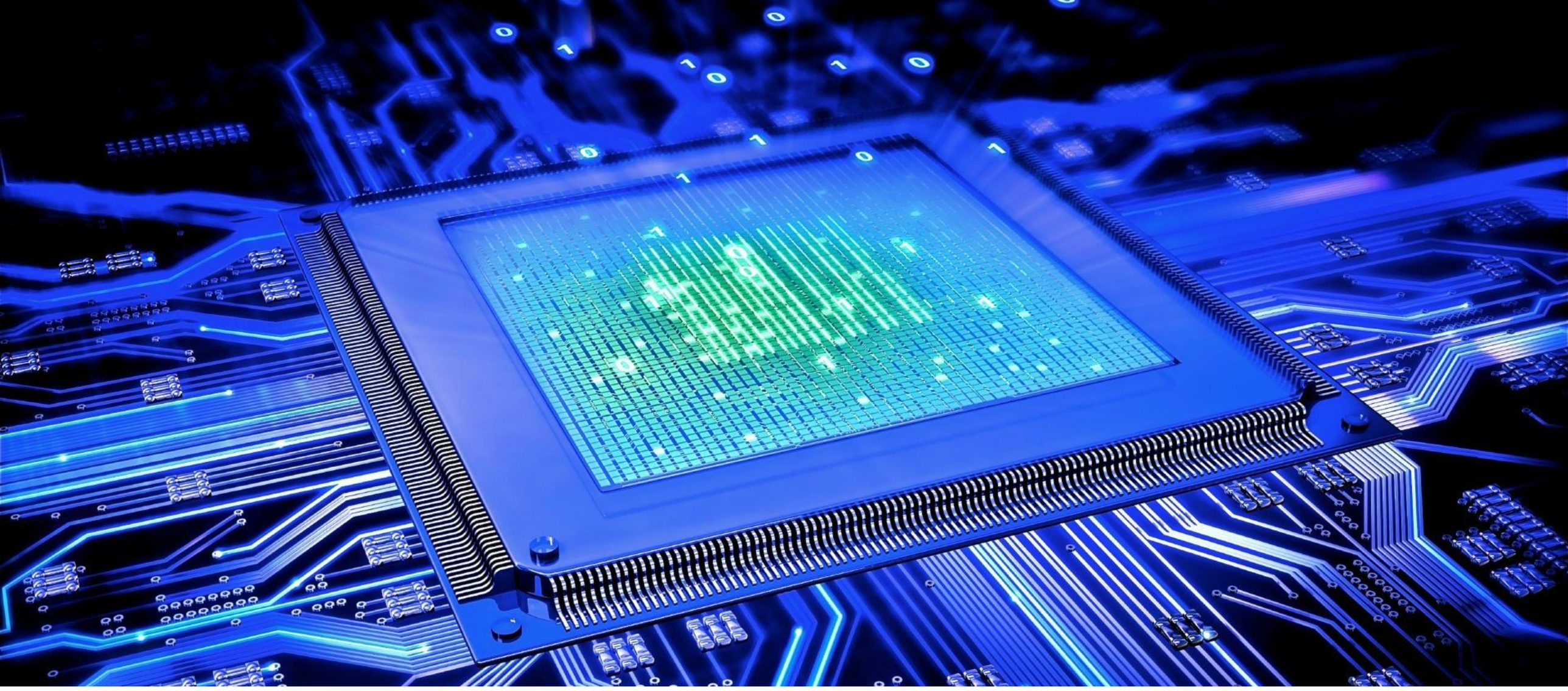


CD4002/CN4002 Computer Systems and Networks

Week 3 CPU and Memory

University of East London

Pioneering Futures Since 1898

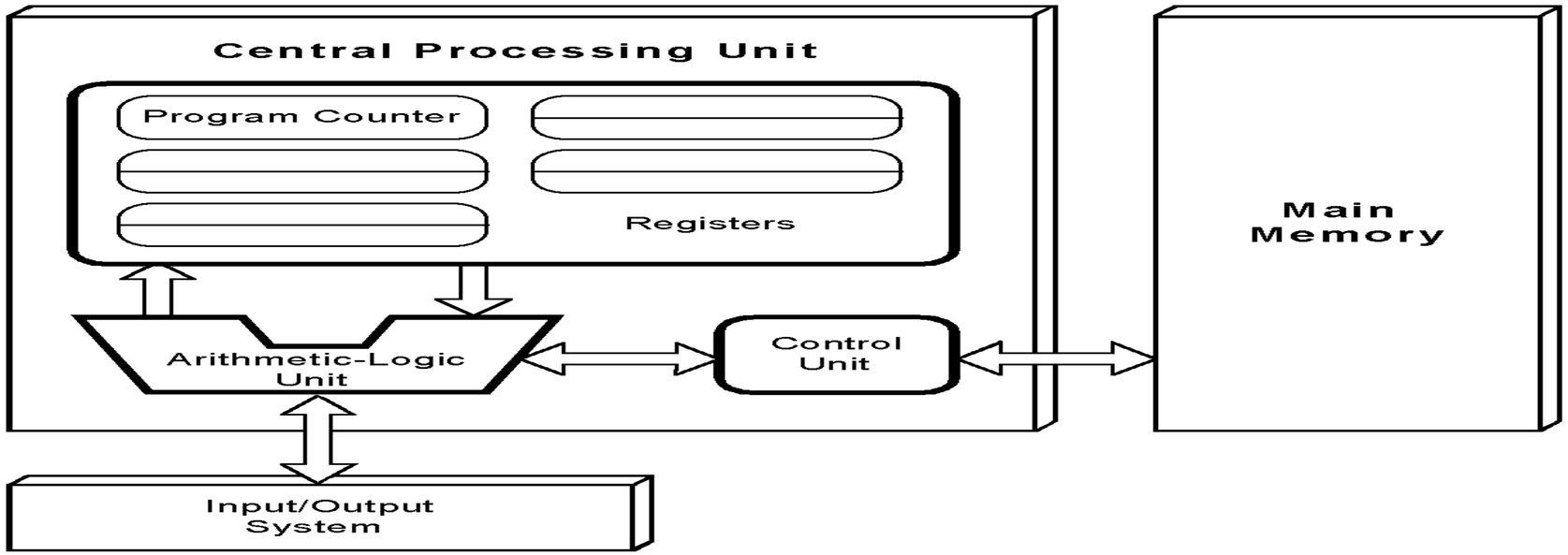
Agenda

- The Little Man Computer (LMC)
- The components of the CPU
- The memory subsystem
- Memory implementations

Program (Instructions) Execution in von Neumann Model

The Von Neumann computer systems use the fetch-decode-execute cycle to run programs as follows:

1. The **Control Unit** fetches the next instruction from **memory** using the **program counter PC** to determine where the instruction is located.
2. The instruction is decoded into a language that the **ALU** can understand.
3. The **data** operands required to execute the instruction are fetched from **memory** and placed into **registers** within the CPU.
4. The **ALU** executes the instruction and places results in **registers or memory.**



Load Fetch/Execute Cycle in Computer Systems

1. PC -> MAR	Transfer the address from the PC to the MAR
2. MDR -> IR	Transfer the instruction to the IR
3. IR(address) -> MAR	Address portion of the instruction loaded in MAR
4. MDR -> A	Actual data copied into the accumulator
5. PC + 1 -> PC	Program Counter incremented

The Little Man Computer (LMC)

The Little Man Computer (LMC) is a model used to simulate the operation of the computer.

Mailboxes

- Hold three-digit numbers. The individual mailboxes are identified by consecutive **addresses**, numbered 00-99.
- The **contents** of each individual mailbox may represent either **data** or **instructions (** as in the von Neumann's stored program)

Calculator

- Used is used to temporarily hold numbers, and to add and subtract numbers.

Instruction location counter

- Holds a two-digit number that can be incremented by clicking, or reset from outside the mailroom

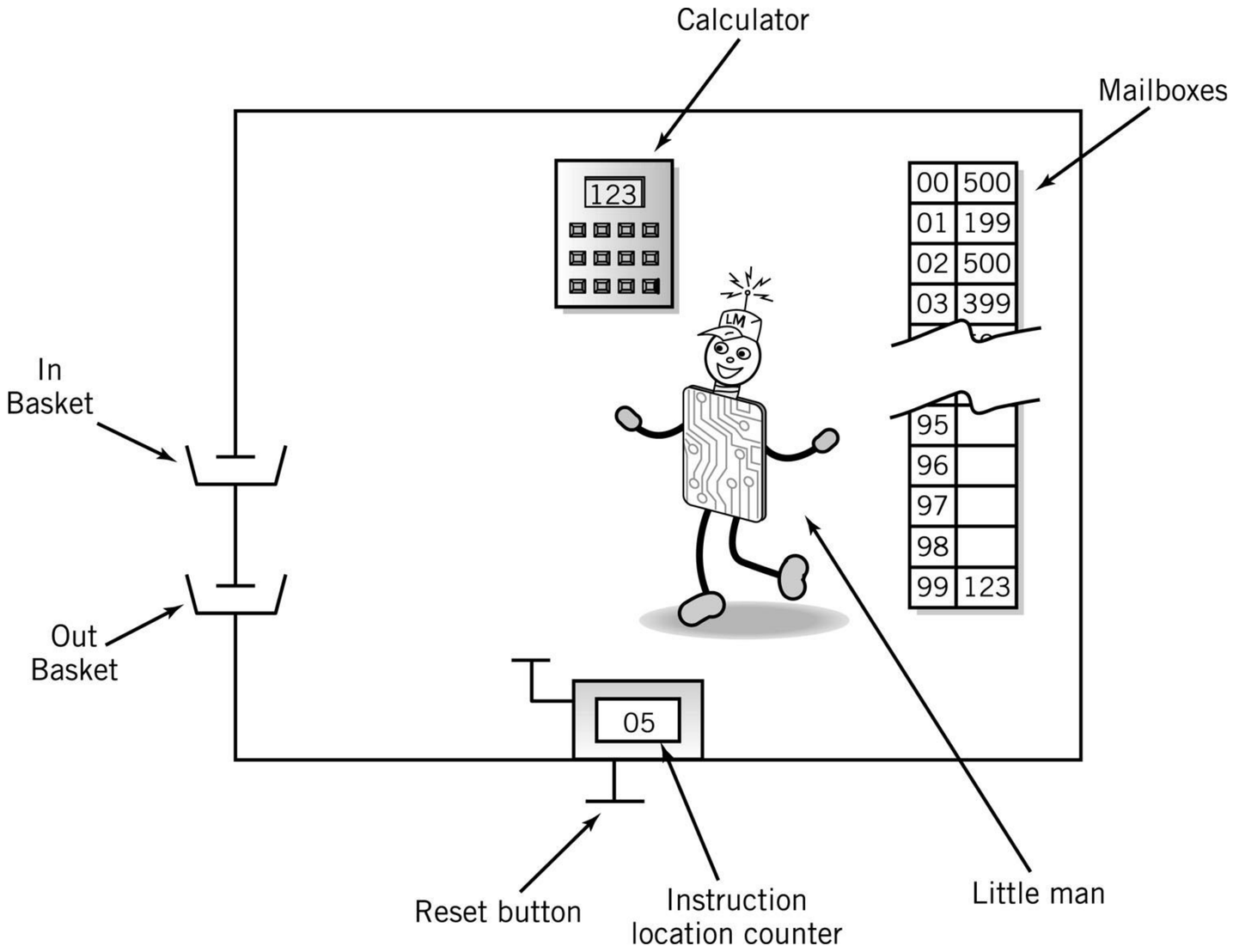
The Little Man

- Performs all the tasks in the mailroom, such as using the calculator, reading the numbers in the in basket, or incrementing the counter.

In and out baskets

- Enables the Little Man to interact with the outside world – communication between the Little Man and a user is achieved by placing three-digit numbers in the in and out baskets.

The Little Man Computer

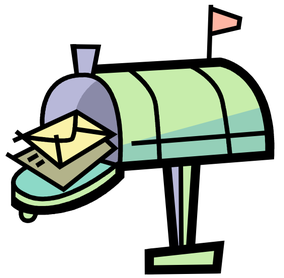


The Mailboxes: Holds the Address vs. Content

- Addresses are consecutive

- Content may be
  - Data or
  - Instructions

Address	Content

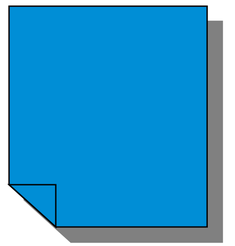


The Content: holds the Instructions

- Op code
  - Operation code which uniquely identifies the instruction
  - A mnemonic (a shortened form of the instruction name)

- Operand
  - Object to be manipulated
    - Data or
    - Address of data

Address	Content	
	Op code	Operand



The Instruction Set consist of following operation:

Arithmetic		
	1xx	ADD
	2xx	SUBTRACT
Data Movement		
	3xx	STORE
	5xx	LOAD
Input/Output		
	901	INPUT
	902	OUTPUT
Machine Control		
	000	HALT
		COFFEE BREAK

For Arithmetic Instructions

1. Read mailbox
2. Perform operation in the calculator

	Content	
	Op Code	Operand (address)
ADD	1	xx
SUB (subtract)	2	xx

Data Movement Instructions

- Between mailbox and calculator

	Content	
	Op Code	Operand (address)
STO (store)	3	xx
LDA (load)	5	xx

Input/Output Instructions

- Move data between calculator and in/out baskets

	Content	
	Op Code	Operand (address)
INP (input)	9	01
OUT (output)	9	02

For Data storage locations

- Physically identical to mailboxes containing instructions
- Located in mailboxes beyond sequence of instructions
- Identified by ***DAT*** mnemonic e.g. DAT 001

Quiz Time

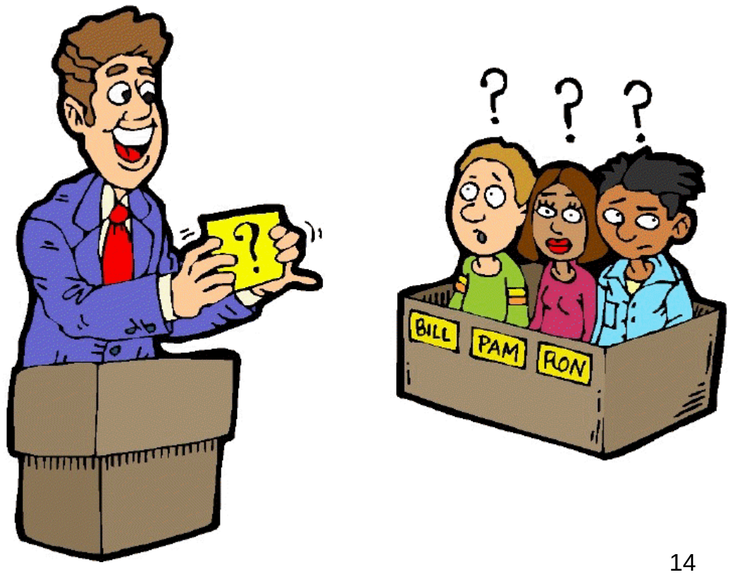
Q. What does the LMC instruction 199 (ADD 99) do?

a. Adds the contents of mailbox 99 to the calculator.

b. Adds the contents of the calculator to mailbox 99.

c. Adds the value 99 to the calculator.

d. Adds the value 99 to mailbox 99.



Quiz Time

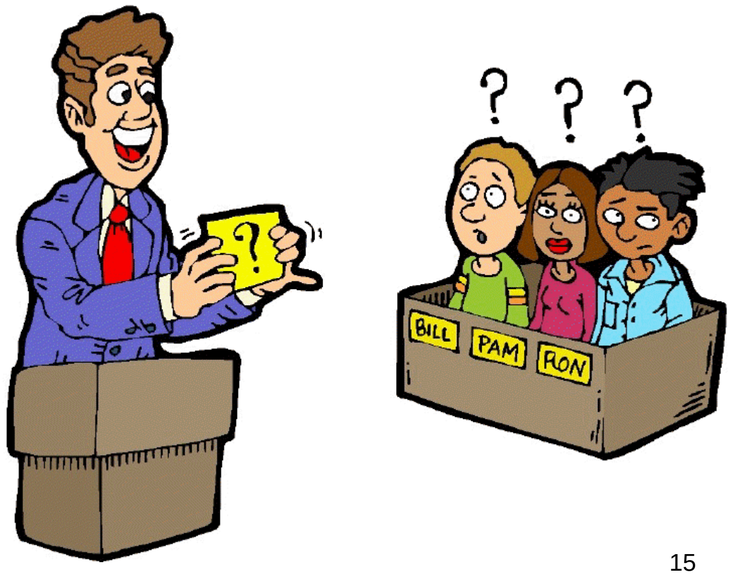
Q. What does the LMC instruction 398 (STO 98) do?

a. Stores the contents of mailbox 98 in the calculator.

b. Stores the contents of the calculator in mailbox 98.

c. Stores the value 98 in the calculator.

d. Stores the value 98 in mailbox 98.



Quiz Time

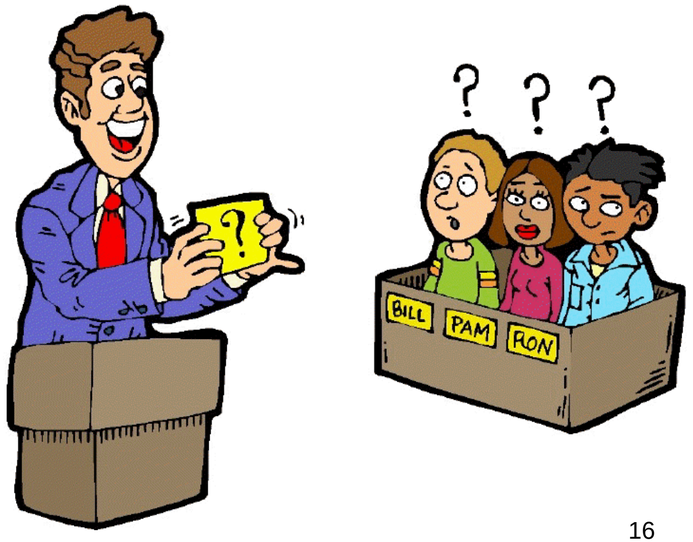
Q. Executing the instruction 297 (SUB 97) will

a. Update the contents of the calculator and mailbox 97

b. Update the contents of the calculator but not the contents of mailbox 97

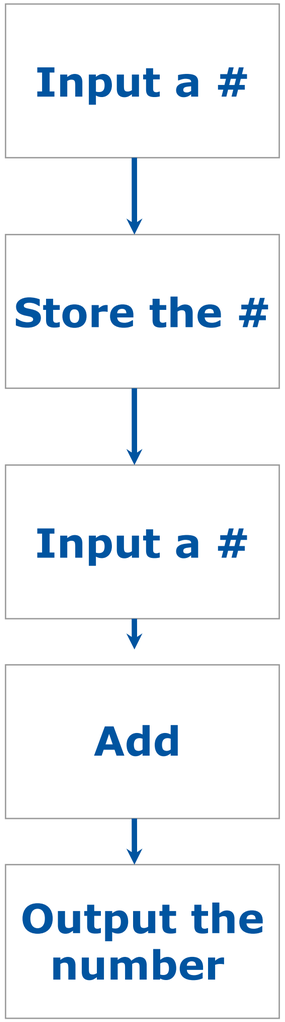
c. Update the contents of mailbox 97 but not the calculator

d. Update neither the calculator nor mailbox 97.



Simple Program 1: Add 2 Numbers

- Assume data is stored in mailboxes with addresses >90
- Now let's write the instructions



Program to Add 2 Numbers Using Mnemonics

Mailbox	Mnemonic	Instruction Description
00	INP	;input 1st Number
01	STO 99	;store number @ loc 99
02	IN	;input 2nd Number
03	ADD 99	;add 1st # to 2nd #
04	OUT	;output result
05	COB	;stop
99	DAT 000	;data

Program to Add 2 Numbers Using Machine Code

Mailbox	Code	Instruction Description
00	901	;input 1st Number
01	399	;store number @ loc 99
02	901	;input 2nd Number
03	199	;add 1st # to 2nd #
04	902	;output result
05	000	;stop
99	000	;data

What does this program do?

Mailbox	Code	Instruction Description
00	INP	?
01	ADD 90	?
02	OUT	?
03	HLT	?
90	100	?

If the input = 1, what will the output be?

Program Control

- Branching (executing an instruction out of sequence)
  - Changes the address in the instruction location counter
- Halt

	Content	
	Op Code	Operand (address)
BR (Jump)	6	xx
BRZ (Branch on 0)	7	xx
BRP (Branch on +)	8	xx
COB (stop)	0	(ignore)

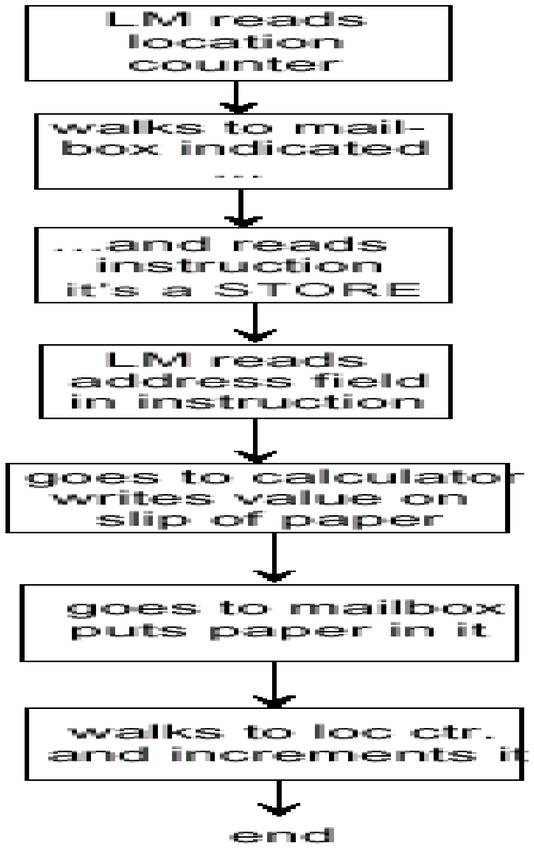
The Complete LMC Instruction Set

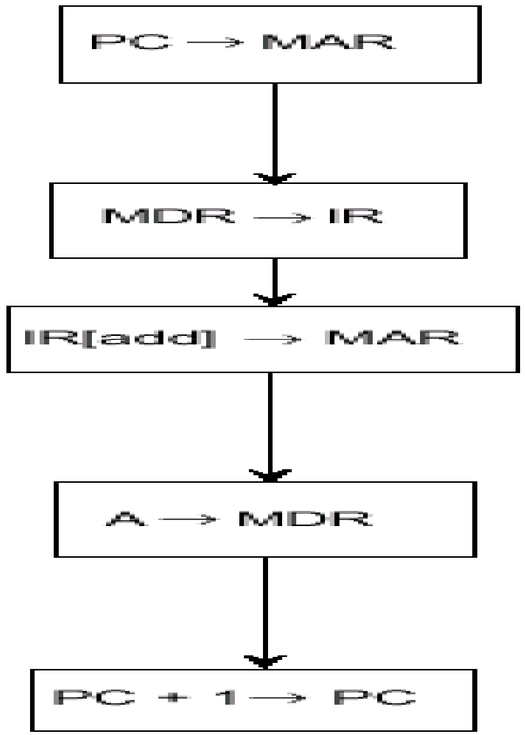
Arithmetic	1xx	ADD
	2xx	SUB
Data Movement	3xx	STORE
	5xx	LOAD
BR	6xx	JUMP
BRZ	7xx	BRANC ON 0
BRP	8xx	BRANCH ON +
Input/Output	901	INPUT
	902	OUTPUT
Machine Control (coffee break)	000	HALT COB

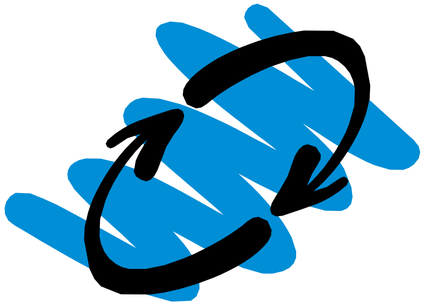
The only instructions that the Little Man understands!

Instruction Cycle:- LMC vs. CPU Fetch and Execute Cycle

- Fetch: Little Man finds out what instruction he must execute
- Execute: Little Man performs the work required by the instruction.

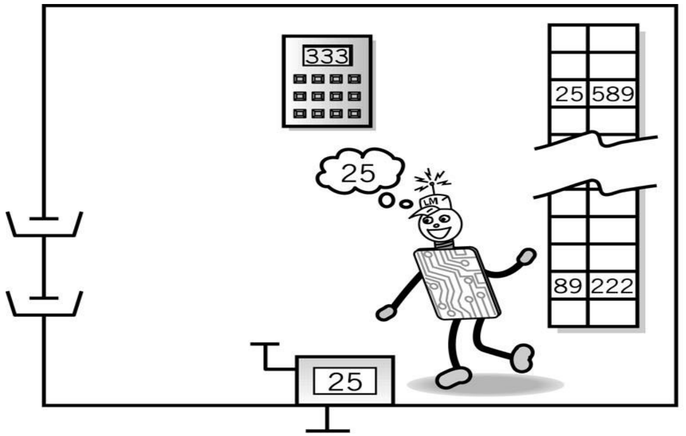




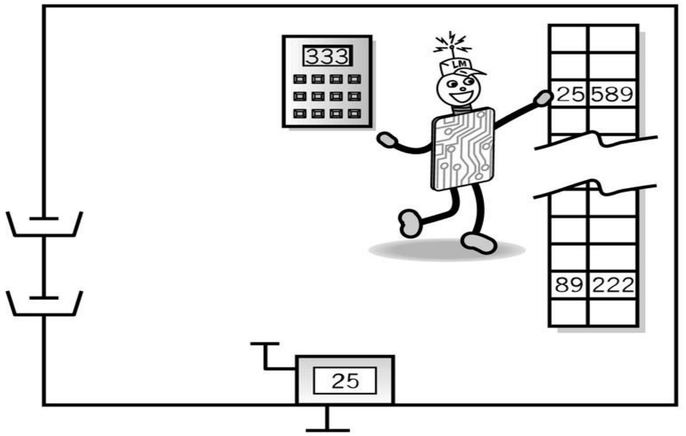


Fetch Portion of the Fetch Execute Cycle

1. Little Man reads the address from the instruction location counter

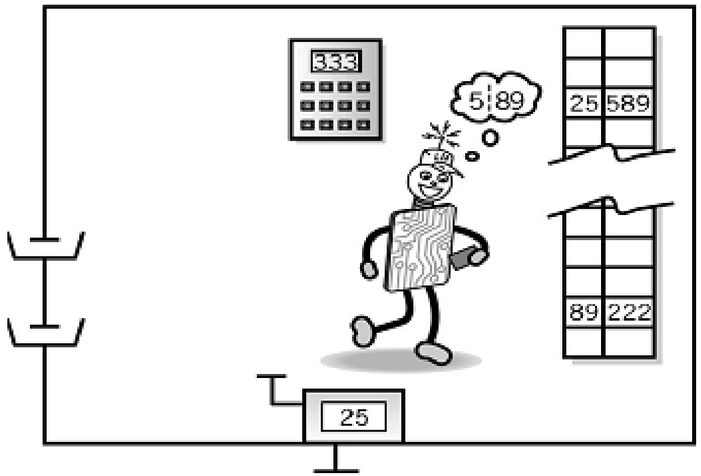


2. He walks over to the mailbox that corresponds to the counter



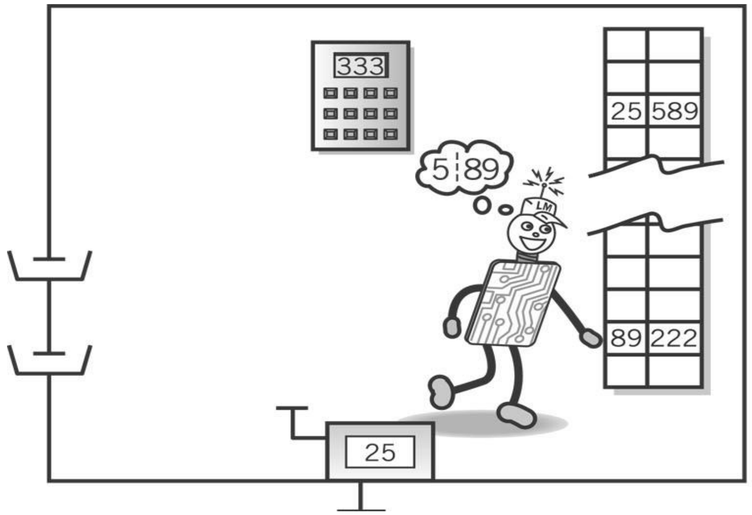
Fetch (cont.)

3. And reads the number on the slip of paper (he puts the slip back in case he needs to read it again later)

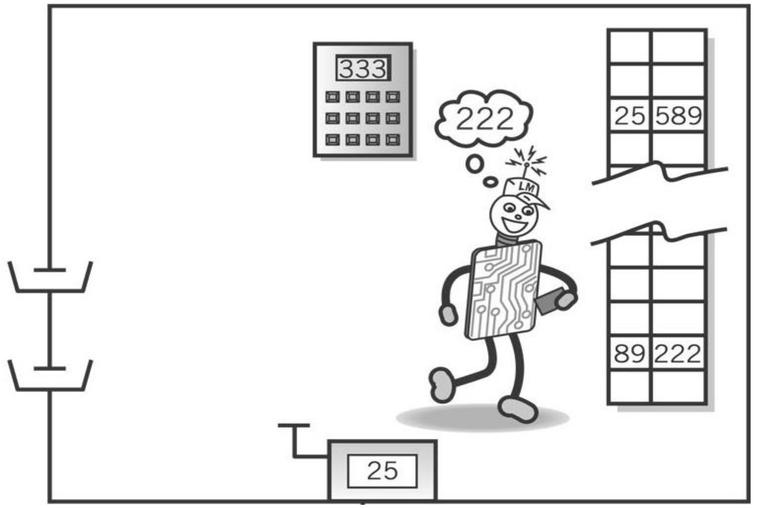


Execute Portion

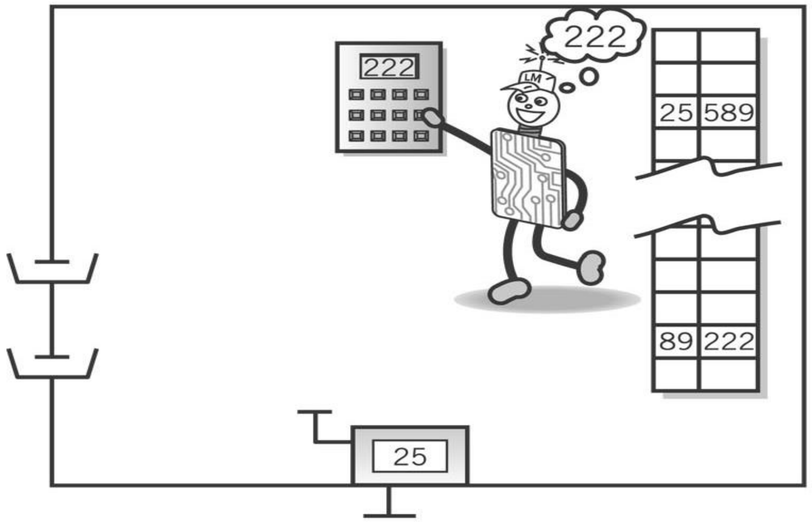
1. The Little Man goes to the mailbox address specified in the instruction he just fetched.



2. He reads the number in that mailbox (he remembers to replace it in case he needs it later).

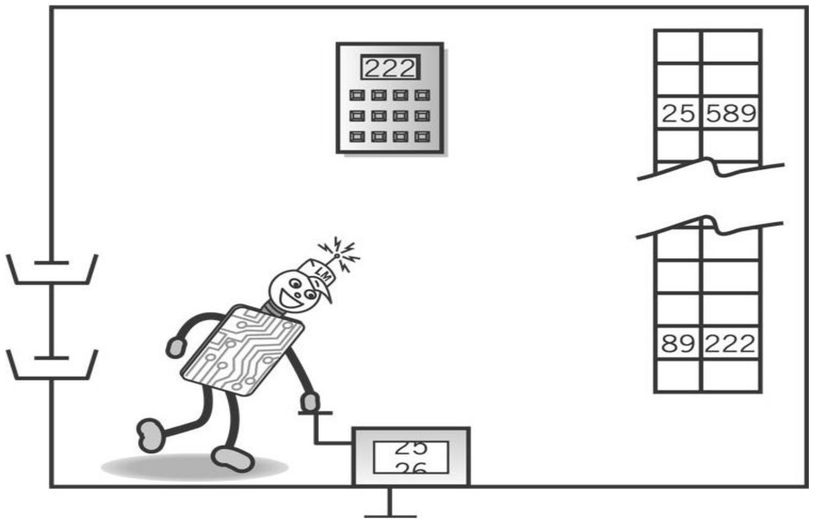


Execute (cont.)



3. He walks over to the calculator and punches the number in.

4. He walks over to the location counter and clicks it, which gets him ready to fetch the next instruction.



CPU: Major Components

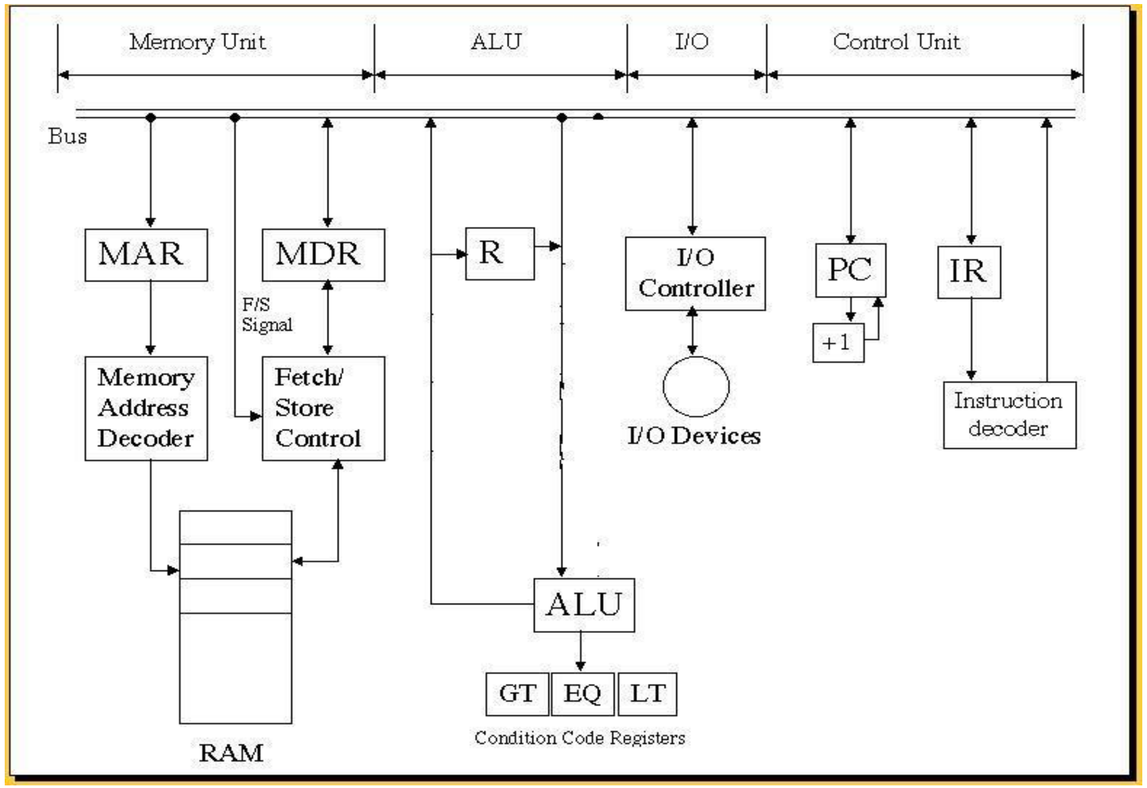
- ALU (arithmetic logic unit)
  - Performs calculations and comparisons
- CU (control unit): performs fetch/execute cycle
  - Functions:
    - Moves data to and from CPU registers and other hardware components
    - Accesses program instructions and issues commands to the ALU
  - Subparts:
    - Memory management unit: supervises fetching instructions and data
    - I/O Interface: sometimes combined with memory management unit as Bus Interface Unit
- Registers
  - Example: *Program counter (PC)* or instruction pointer determines next instruction for execution

Registers

- Small, permanent but volatile storage locations within the CPU used for a particular purpose
- Manipulated directly by the Control Unit
- Some are wired for a specific function, others are user-visible
- Size in bits or bytes (not GB like memory)
- Can hold data, an address or an instruction

Special-Purpose Registers

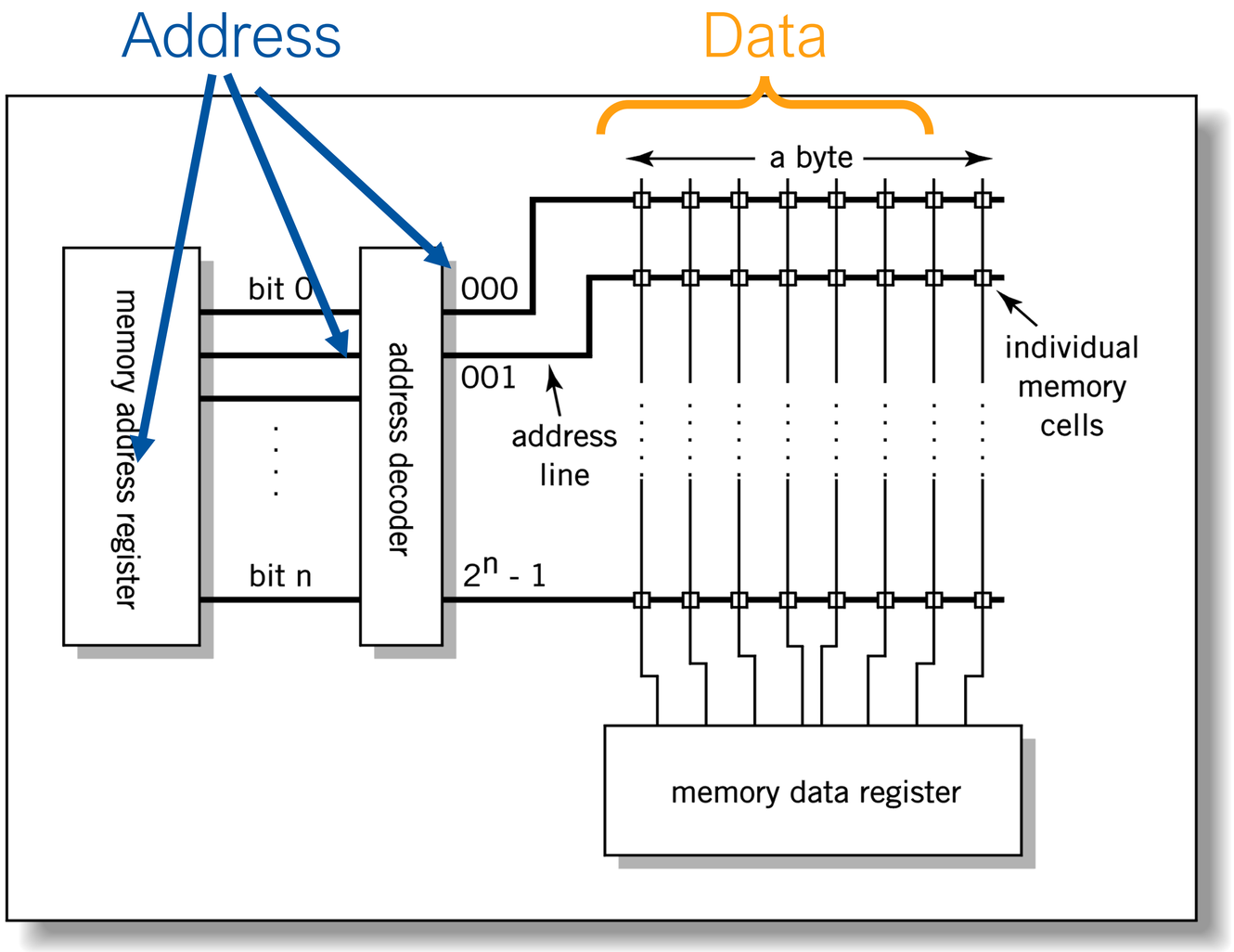
- Program Counter (PC)
  - Also called instruction pointer
- Instruction Register (IR)
  - Stores instruction fetched from memory
- Memory Address Register (MAR)
- Memory Data Register (MDR)
- Status Registers
  - Status of CPU and currently executing program
  - Flags (one bit Boolean variable) to track condition like arithmetic carry and overflow, power failure, internal computer error



Fill in the blanks

CPU Component	LMC Component
Arithmetic and logic unit (ALU)	?
Control unit (CU)	?
Program counter	?
I/O interface	?

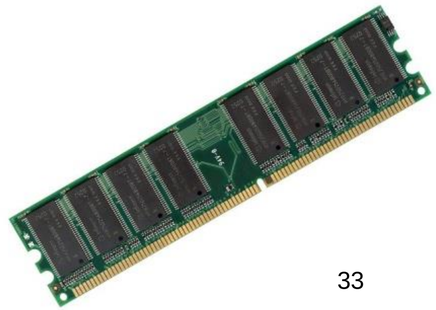
Relationship between MAR, MDR and Memory



RAM: Random Access Memory

- DRAM (Dynamic RAM)
  - Most common, cheap
  - Volatile: must be refreshed (recharged with power) 1,000’s of times per second

- SRAM (static RAM)
  - Faster than DRAM but more expensive
  - Volatile but does not need refreshing
  - Small amounts frequently used in cache memory for high-speed access

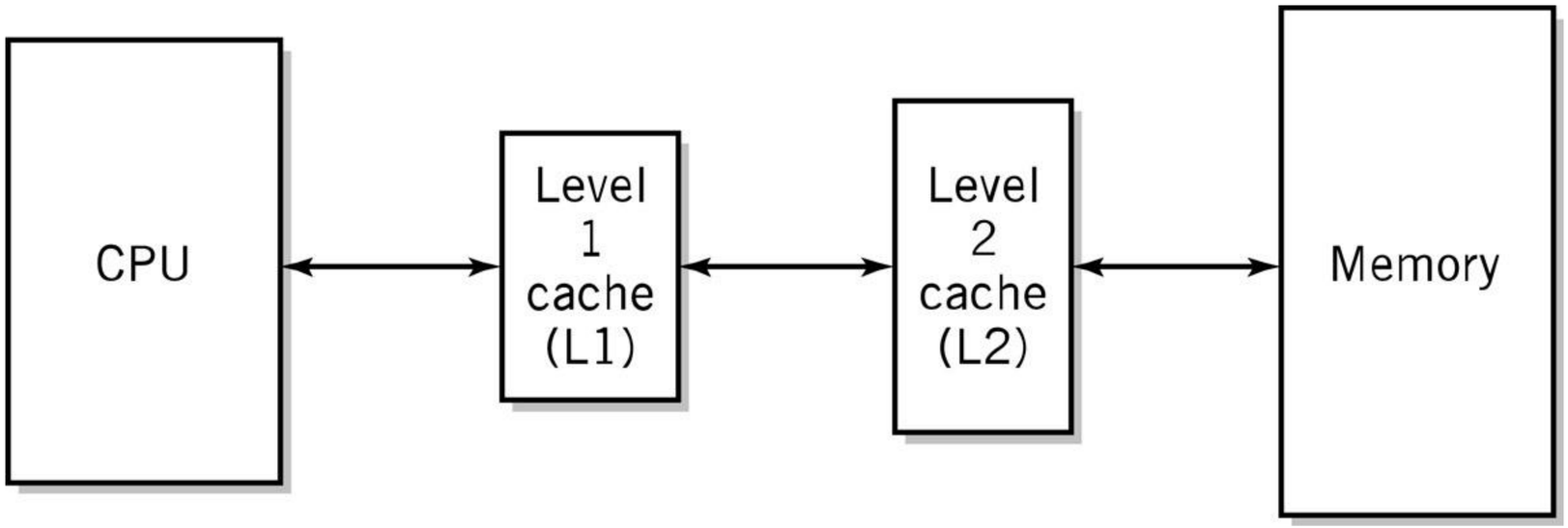


Cache Memory

- Even the fastest hard disk has an access time measured in milliseconds
- A 3Ghz CPU waiting 1 millisecond wastes 3 million clock cycles!
- Hit ratios of 90% common
- 50%+ improved execution speed
- Locality of reference is why caching works
  - Most memory references confined to small region of memory

Two-level Caches

- Why do the sizes of the caches have to be different?



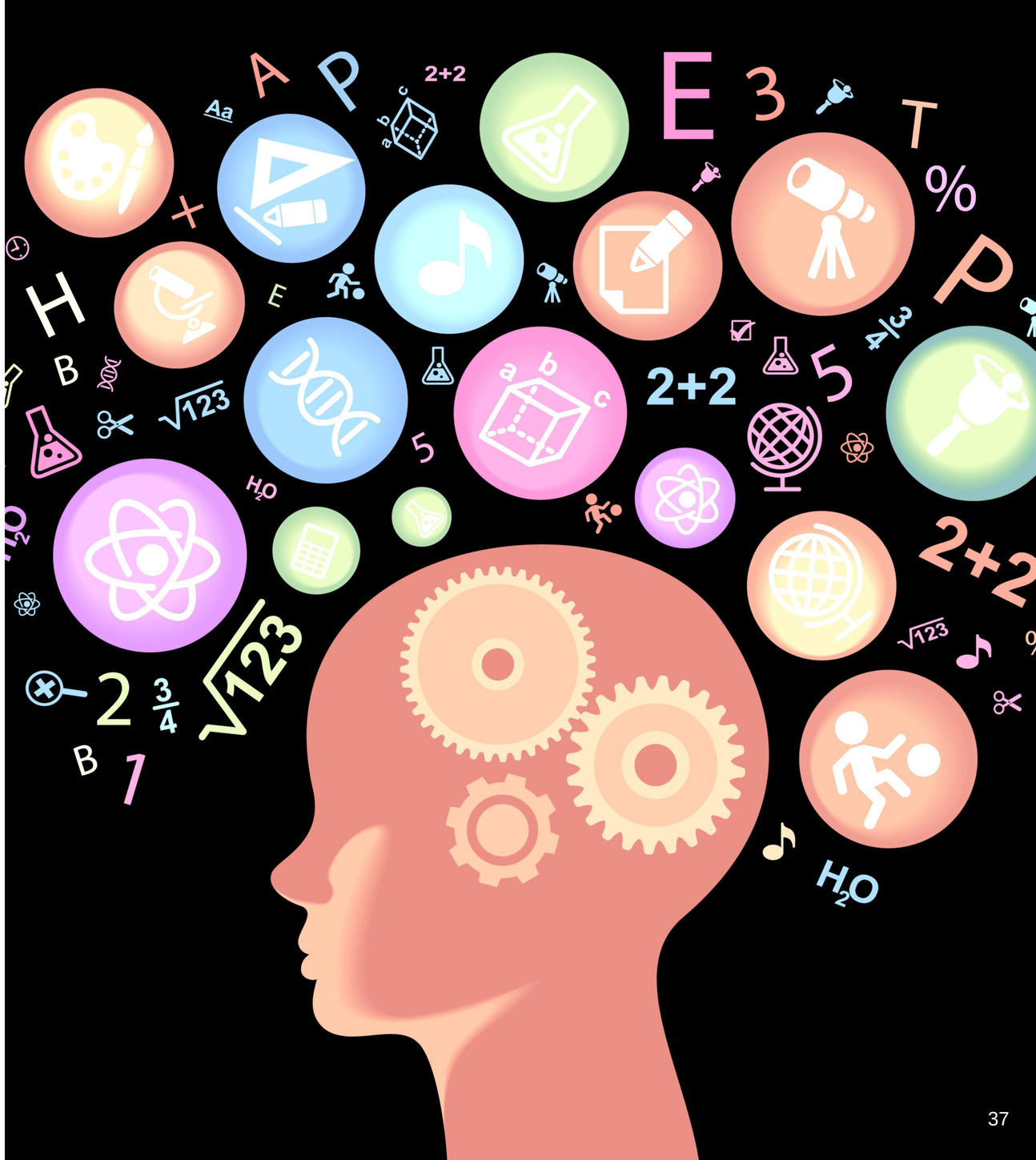
ROM - Read Only Memory

- Non-volatile memory to hold software that is not expected to change over the life of the system a.k.a. firmware
- EEPROM
  - Electrically Erasable Programmable ROM
  - Traditionally used to store system BIOS
- Flash memory
  - Based on EEPROM technology
  - Designed for high speed and density
  - Used for BIOS/UEFI, USB flash drives, SSDs, digital cameras, mobile phones etc.
  - Cannot be updated indefinitely

Learning Objectives

On completion of this topic, you will be able to:

- Explain the concept of an instruction set
- Identify the stages of the fetch execute cycle
- Identify the components of a real CPU and describe their functions
- Distinguish between the various types of memory and their use



Directed Reading

- Englander, Chapters 6 and 7 (up to and including 7.3)
- Stallings, Chapters 4, 5 and 12