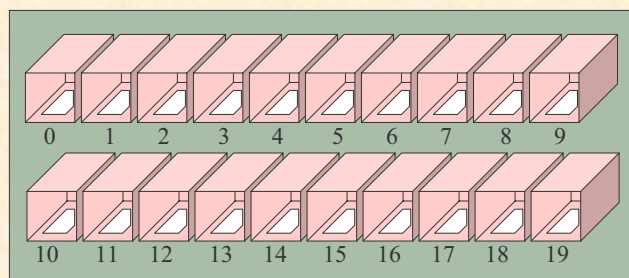


## HOW A COMPUTER WORKS - GETTING STARTED

- ❑ A **computer** is a device for solving problems
  - its advantage is its speed
- ❑ **Programming** is nothing more than planning, scheduling, or performing a task or an event
- ❑ **Computer Programming** is the process of planning a sequence of steps for a computer to follow:
  - you can't do this unless you know how to solve the problem yourself
- ❑ A **Computer Program** is a list of instructions to be performed by a computer
- ❑ **Steps** in using a computer:
  - understand the problem to be solved
  - figure out how to solve it yourself
  - plan a solution to solve the problem
  - write a program to tell the computer how to solve the problem

## SIMULATING A TOY COMPUTER

- ❑ We will simulate a toy computer to understand how one works.
- ❑ First, we need a wall of mailboxes each with a unique number as you might find in a post office (or the mail room in your dorm).
  - each mailbox contains a piece of paper large enough to write one number on it



## SIMULATING A TOY COMPUTER (CONTINUED)

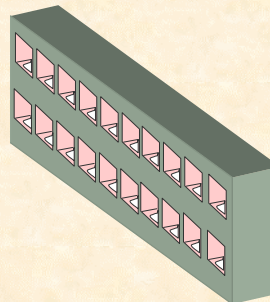
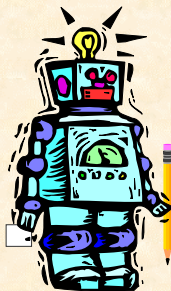
- Suppose that the mailboxes are in a small room with a robot.
- The robot holds a piece of paper in one hand and a pencil in the other.
  - the paper is large enough for only one number to be written on it
- The robot knows how to:
  - add two numbers
  - subtract two numbers
  - multiply two numbers
  - divide two numbers
  - compare two numbers to determine which is bigger
  - read the number on the paper in a specified mailbox
  - write a number on the paper in a specified mailbox
  - read or write a number on the paper in its hand
  - go to sleep
  - wake up
  - say the number written on the paper in its hand or in a mailbox

Programming I

Lecture 1 -- 3

## SIMULATING A TOY COMPUTER (CONTINUED)

- You give commands to the robot one at a time by speaking into a microphone.
  - the robot does not remember anything from one command to the next
  - anything that will be needed in the future must be written on a piece of paper (either in a mailbox or in the robot's hand)
- Initially, the robot is asleep.
  - The papers in the mailboxes and the robot's hand are all blank



Programming I

Lecture 1 -- 4

## COMPUTE THE SUM AND PRODUCT OF THREE NUMBERS

### □ Assumptions

- we pick the three numbers to use
- robot computes the sum first and says what it is
- robot computes the product next and says what it is

### □ Solution - first attempt

- wake up the robot
- robot writes the three numbers in three mailboxes
- robot computes the sum of the three numbers
- robot says what the computed sum is
- robot computes the product of the three numbers
- robot says what the computed product is
- robot goes to sleep

- Need to refine this solution to add more details since the robot won't understand some of these commands.

## REFINED SOLUTION TO COMPUTE SUM AND PRODUCT

### □ Solution -- second attempt

- wake up the robot
- robot writes the three numbers in three mailboxes
  - pick a number and tell robot to write it in mailbox 1
  - pick another number and tell robot to write it in mailbox 2
  - pick a third number and tell robot to write it in mailbox 3
- robot computes the sum of the three numbers
  - write 0 on paper in hand
  - read number in mailbox 1, add it to number on paper in hand, and write the sum on paper in hand
  - read number in mailbox 2, add it to number on paper in hand, and write the sum on paper in hand
  - read number in mailbox 3, add it to number on paper in hand, and write the sum on paper in hand
- robot says what the computed sum is
  - say number on paper in hand

## REFINED SOLUTION (CONTINUED)

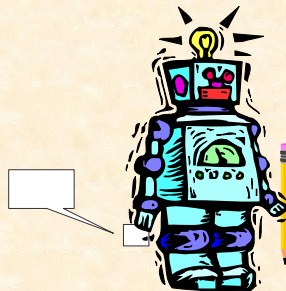
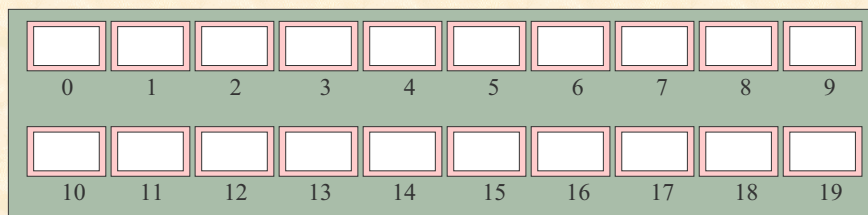
### □ Solution -- second attempt (continued)

- robot computes the product of the three numbers
  - write 1 on paper in hand
  - read number in mailbox 1, multiply it by number on paper in hand, and write the product on paper in hand
  - read number in mailbox 2, multiply it by number on paper in hand, and write the product on paper in hand
  - read number in mailbox 3, multiply it by number on paper in hand, and write the product on paper in hand
- robot says what the computed product is
  - say number on paper in hand
- robot goes to sleep

□ The robot will understand these commands, so no more refinement is needed.

## TOY COMPUTER SIMULATION OF THE SOLUTION

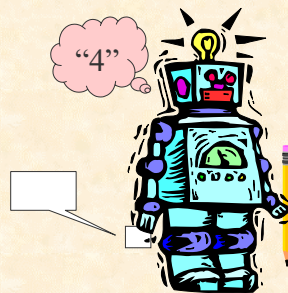
### □ wake up the robot



## TOY COMPUTER SIMULATION OF THE SOLUTION

- pick a number and tell robot to write it in mailbox 1 (pick 4)

	4								
0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19



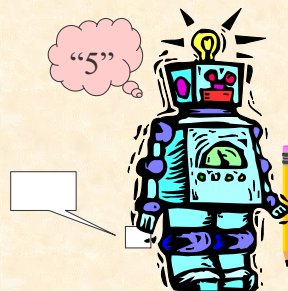
Programming 1

Lecture 1 -- 9

## TOY COMPUTER SIMULATION OF THE SOLUTION

- pick another number and tell robot to write it in mailbox 2 (pick 5)

	4	5							
0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19



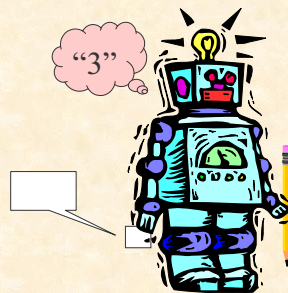
Programming 1

Lecture 1 -- 10

## TOY COMPUTER SIMULATION OF THE SOLUTION

- pick a third number and tell robot to write it in mailbox 3 (pick 3)

	4	5	3						
0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19



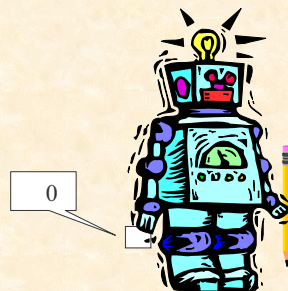
Programming 1

Lecture 1 -- 11

## TOY COMPUTER SIMULATION OF THE SOLUTION

- write 0 on paper in hand

	4	5	3						
0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19

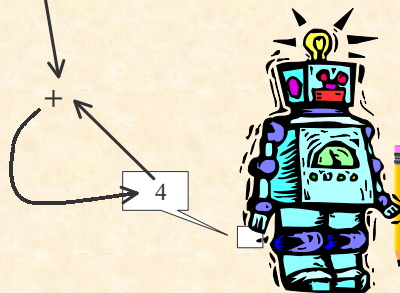
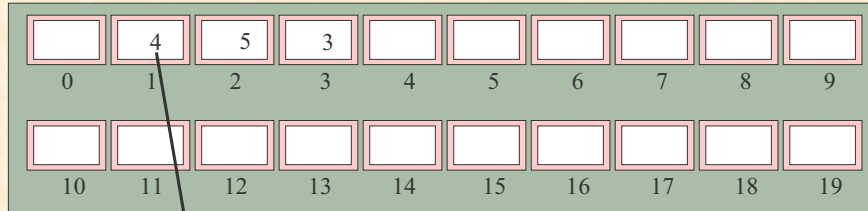


Programming 1

Lecture 1 -- 12

## TOY COMPUTER SIMULATION OF THE SOLUTION

- read number in mailbox 1, add it to number on paper in hand, and write the sum on paper in hand

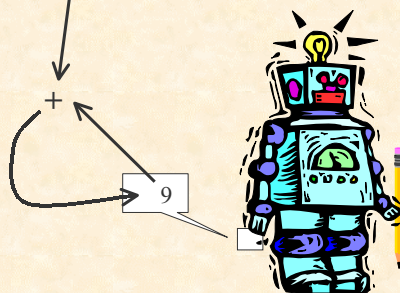
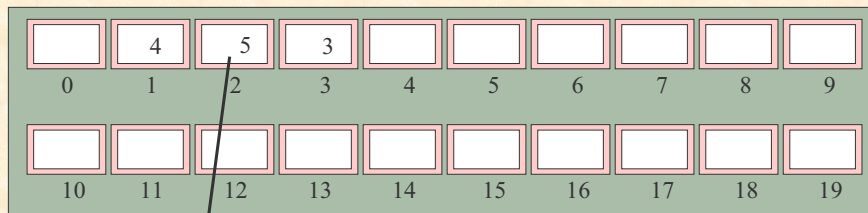


Programming 1

Lecture 1 -- 13

## TOY COMPUTER SIMULATION OF THE SOLUTION

- read number in mailbox 2, add it to number on paper in hand, and write the sum on paper in hand

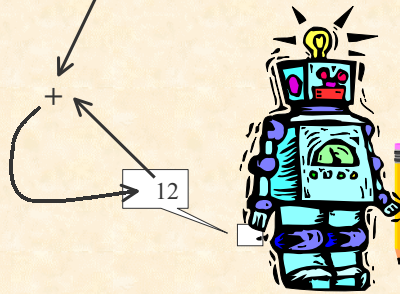
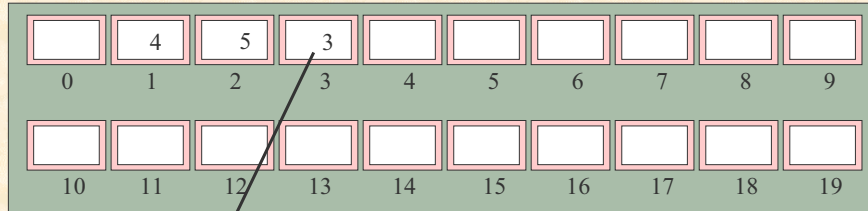


Programming 1

Lecture 1 -- 14

## TOY COMPUTER SIMULATION OF THE SOLUTION

- read number in mailbox 3, add it to number on paper in hand, and write the sum on paper in hand

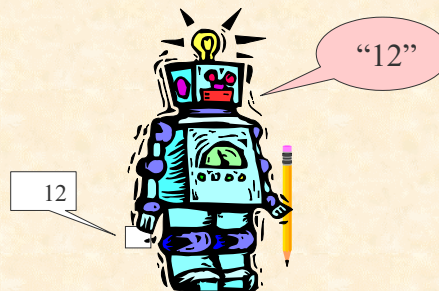
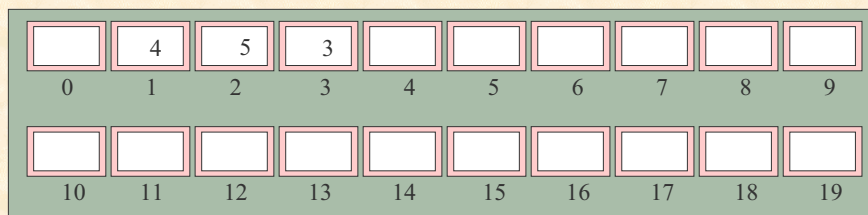


Programming I

Lecture 1 -- 15

## TOY COMPUTER SIMULATION OF THE SOLUTION

- say number on paper in hand

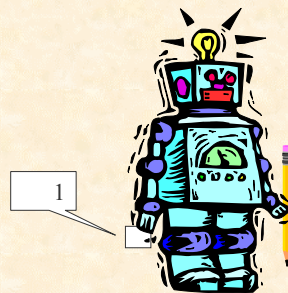
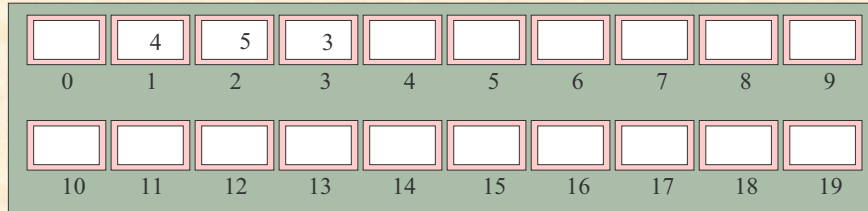


Programming I

Lecture 1 -- 16

## TOY COMPUTER SIMULATION OF THE SOLUTION

- write 1 on paper in hand

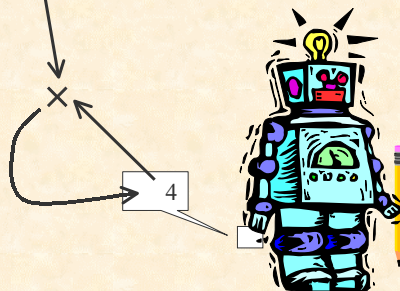
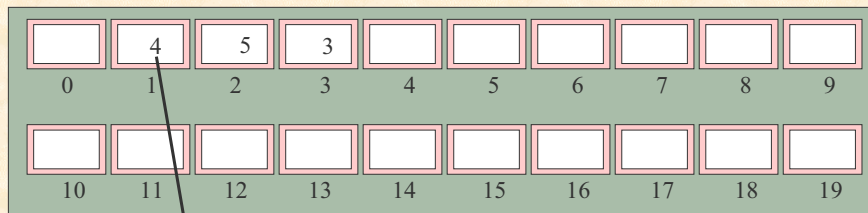


Programming 1

Lecture 1 -- 17

## TOY COMPUTER SIMULATION OF THE SOLUTION

- read number in mailbox 1, multiply it by number on paper in hand, and write the product on paper in hand

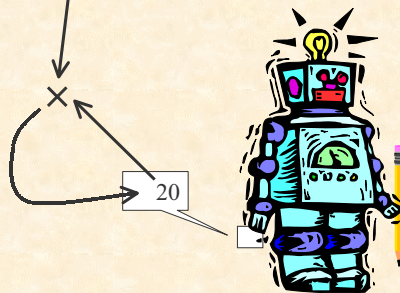
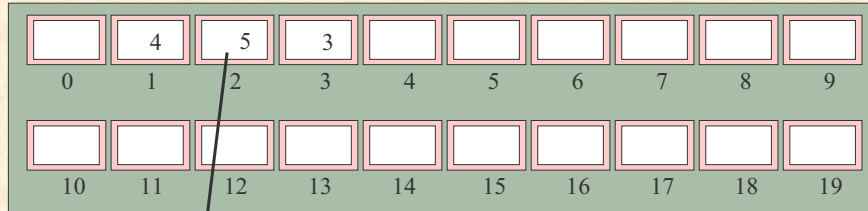


Programming 1

Lecture 1 -- 18

## TOY COMPUTER SIMULATION OF THE SOLUTION

- read number in mailbox 2, multiply it by number on paper in hand, and write the product on paper in hand

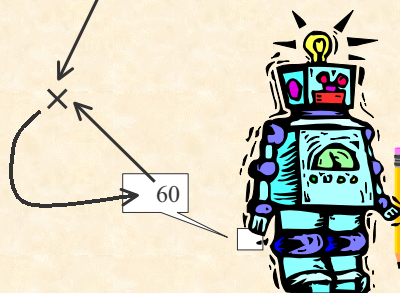
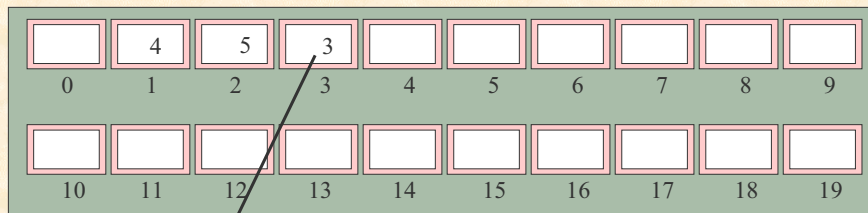


Programming 1

Lecture 1 -- 19

## TOY COMPUTER SIMULATION OF THE SOLUTION

- read number in mailbox 3, multiply it by number on paper in hand, and write the product on paper in hand

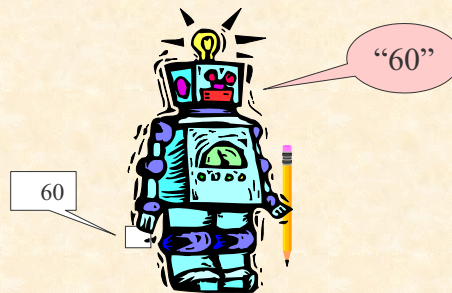
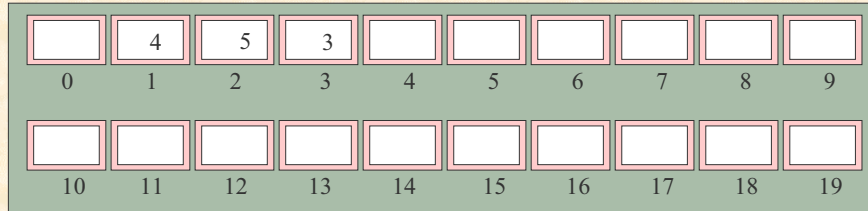


Programming 1

Lecture 1 -- 20

## TOY COMPUTER SIMULATION OF THE SOLUTION

- say number on paper in hand

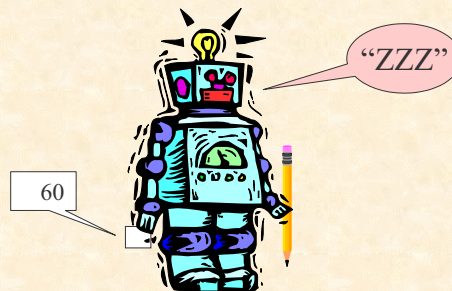
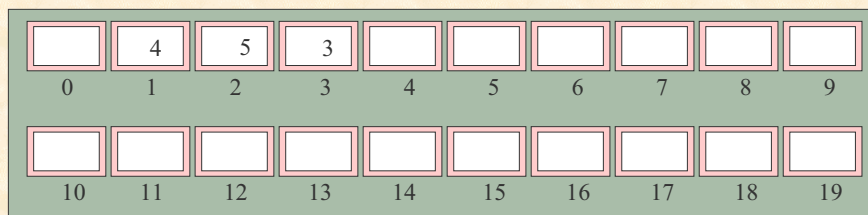


Programming 1

Lecture 1 -- 21

## TOY COMPUTER SIMULATION OF THE SOLUTION

- robot goes to sleep



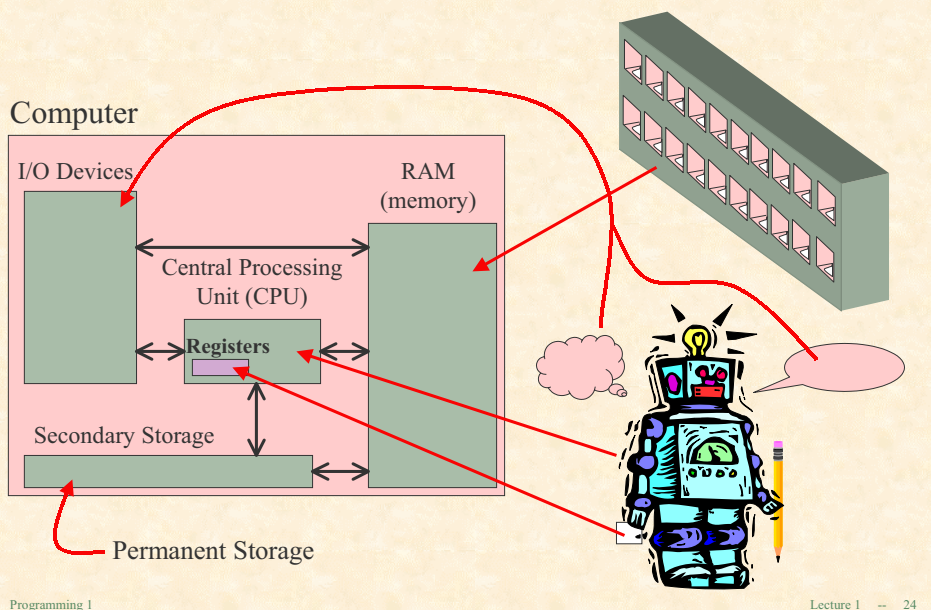
Programming 1

Lecture 1 -- 22

## THINGS TO REMEMBER

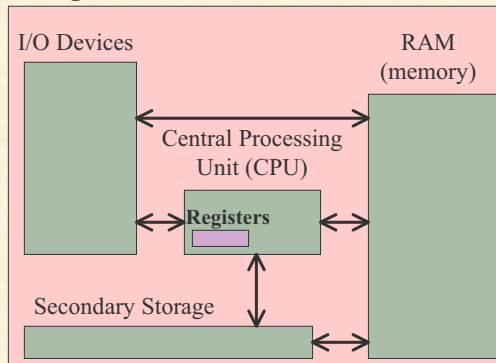
- ❑ The robot in our toy computer has a small set of commands that it understands and can do.
  - arithmetic operations
  - comparisons between two numbers
  - read and write number in a mailbox and on paper in its hand
  - say a number in a mailbox or on paper in its hand
  - hear a number given (spoken) to it
  - go to sleep and wake up
- ❑ A piece of paper can have only one number written on it at a time.
  - old number is erased before the new number is written
- ❑ The robot cannot remember anything from one command to the next.
  - everything that must be remembered must be written on paper
- ❑ The robot will do exactly what it is commanded to do and nothing more.

## ARCHITECTURE OF A COMPUTER



## I/O DEVICES

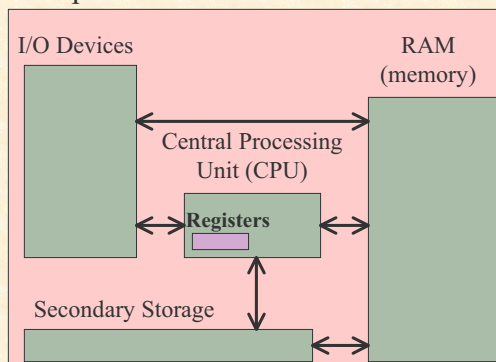
### Computer



- ❑ Communications with the outside world.
- ❑ Input devices:
  - keyboard
  - mouse
  - joystick
- ❑ Output devices:
  - screen
  - printer
  - computer speakers

## CENTRAL PROCESSING UNIT (CPU)

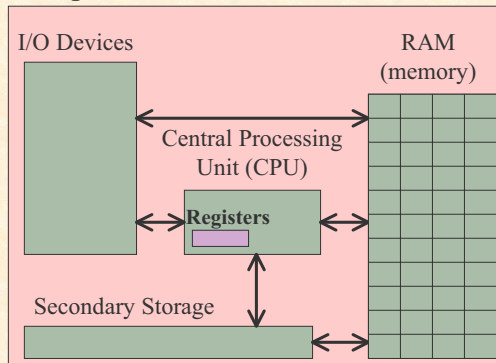
### Computer



- ❑ Execute the instructions of a program.
  - Arithmetic operations
  - comparison operations
  - bookkeeping
- ❑ Uses one or more registers as scratch space for storing numbers between instructions.
- ❑ A typical CPU today can execute millions of arithmetic operations in a second.

## MAIN MEMORY

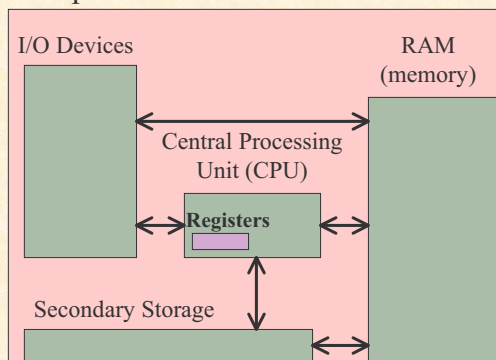
### Computer



- ❑ Sometimes called random access memory (RAM).
- ❑ Stores the numbers (data) that a program uses when it runs on the computer.
- ❑ Also stores the instructions of the program that is running on the computer.
- ❑ Divided into fixed size units of memory called words.
  - each word stores one number

## SECONDARY STORAGE

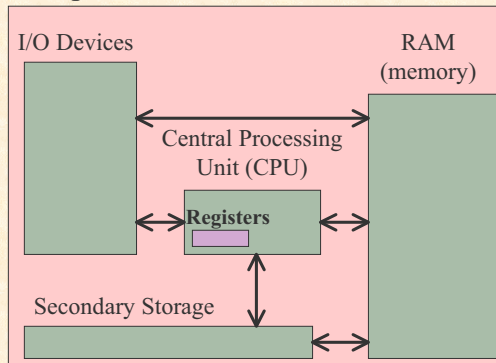
### Computer



- ❑ Permanent storage used to save data and programs when they are not running on the computer.
- ❑ Data and programs are organized into varying size units called files.
  - each file has a unique name
- ❑ Files are organized into directories that can contain subdirectories.
- ❑ Think of secondary storage as a large electronic file cabinet containing folders with files in them.

## PUTTING THE PIECES TOGETHER

### Computer

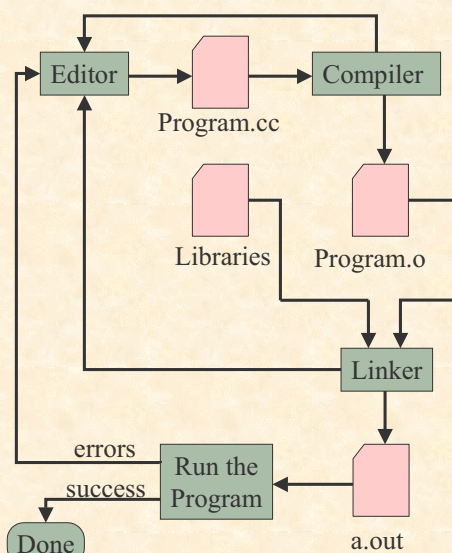


- ❑ A machine language program is created and stored in secondary storage.
- ❑ The program is loaded into memory to run under control of the CPU (and the operating system).
- ❑ The CPU fetches instructions from memory one by one and executes them.
- ❑ Execution of instructions may require reading / writing numbers in CPU registers and in memory
- ❑ The CPU controls I/O devices which move data between memory and the outside world.

## THE PRE-PROGRAMMING PROCESS

- ❑ **Steps** is using a computer to solve a problem:
  - understand the problem to be solved
    - what are the inputs (givens)
    - what are the outputs (desired answers)
    - what are the constraints
  - figure out how to solve it yourself
    - consults other resources as necessary
    - try simple examples by hand
    - THINK !!!!!
  - develop a solution plan to solve the problem
    - verify that it is complete and correct
    - use any convenient notation to write the solution plan
  - write a program to tell the computer how to solve the problem
    - translate your solution plan into a programming language

## THE PROGRAMMING PROCESS



- Use an editor to create a program file (**source** file).
  - contains the text of the program
- Use a compiler to convert the source file into a machine code file (**object** file).
  - convert from "English" to binary with machine operations
- Use a linker to convert the object file into an **executable** file.
  - include other machine code that the program requires
- Run the executable file.
- Iterate from any point as needed.

Programming I

Lecture 1 -- 31

## ROLE OF THE OPERATING SYSTEM

- An operating system controls everything that happens on a computer.
  - interacting with a user
  - running programs
  - sharing and controlling devices for users
- Two popular operating systems are UNIX and Windows.
- Users give commands to the operating system to do things for them.
- Editors, compilers and linkers are part of the operating system.
- An operating system provides services that programs need.
  - getting numbers typed on a keyboard
  - creating a window on the screen
  - displaying text or numbers in a window
- Libraries contain the machine code that implements these services.
  - a linker inserts this machine code into a program when needed

Programming I

Lecture 1 -- 32

## EXAMPLE - DEFINING THE PROBLEM

- ❑ You have just been invited to spend your next vacation biking around Europe. It occurs to you that distances in Europe are measured in kilometers. You decide that it would be convenient to have a program that converts distances in kilometers to distances in miles.
- ❑ What are the inputs ?
  - a distance measured in kilometers
- ❑ What are the expected outputs ?
  - the equivalent distance in miles
- ❑ What are the constraints ?
  - distances input in kilometers will always be integers
  - distances output in miles should be real (float)
  - the program need handle only one distance conversion each time it is run

## EXAMPLE - SOLVING THE PROBLEM

- ❑ Consult other resources as needed.
  - find a conversion formula for kilometers to miles
    - one mile is 1.6093 kilometers (Random House Dictionary)
- ❑ Try simple examples by hand.
  - 10 kilometers
  - if one mile is 1.6093 kilometers, then one kilometer is  $1/1.6093 = 0.6214$  miles
  - thus 10 kilometers is 6.214 miles
- ❑ THINK !!!
  - If  $k$  is a distance in kilometers, then  $k \times 0.6214$  is the equivalent distance in miles
- ❑ Now we can write the solution plan.

## EXAMPLE – WRITING THE SOLUTION PLAN

### □ First version:

- get distance in kilometers from the user
- compute the equivalent distance in miles
- display the distance in miles for the user

### □ Second version:

- get distance in kilometers from the user
  - prompt user to enter distance in kilometers
  - **kilometers** ← distance in kilometers from user
- compute the equivalent distance in miles
  - **miles** ← **kilometers** x 0.6214
- display the distance in miles for the user
  - display **miles** with appropriate message

### □ This is sufficient detail to write the required program.

## EXAMPLE – TRANSLATE THE SOLUTION PLAN TO A PROGRAM

```
// Program to convert a distance in kilometers to an
// equivalent distance in miles.
// Version 1 -- character I/O

// include library code for user I/O operations
#include <iostream.h>

void main()
{
    int            kilometers;    /* distance in kilometers */
    double         miles;        /* equivalent distance in miles */

    /* get distance in kilometers from user */
    cout << "Type a distance in kilometers: ";
    cin >> kilometers;

    /* convert distance in kilometers to distance in miles */
    miles = kilometers * 0.6241;

    /* display equivalent distance in miles to user */
    cout << kilometers << " kilometers is "
         << miles << " miles." << endl;
}
}
```

## EXAMPLE -- TRANSLATE THE SOLUTION PLAN TO A PROGRAM

```
// Program to convert a distance in kilometers to an
// equivalent distance in miles.
//
// Version 2 -- windows I/O

#include "GUIObj.h"
#include "maindrv.h"

void main ()
{
    int kilometers; /* distance in kilometers */
    float miles; /* equivalent distance in miles */
    IntTypeIn IntInpBox; /* window to enter distance in KM */
    OKBox msgBox; /* window to display distance in M */

    /* get distance in kilometers from user */
    kilometers = IntInpBox.GetInt("INPUT",
        "Enter a distance in kilometers");

    /* convert distance in kilometers to distance in miles */
    miles = kilometers * 0.6241;

    /* display equivalent distance in miles to user */
    msgBox.Display(miles);
}
```

Programming I

Lecture 1 -- 37

## EXAMPLE -- TRANSLATE THE SOLUTION PLAN TO A PROGRAM

### LISP

```
Define ((
  (KiloToMiles (Lamda
    (K)
      (Times K
        0.6241)))
  ))

KiloToMiles (10)
KiloToMiles (135)
```

Programming I

Lecture 1 -- 38

## EXAMPLE - TRANSLATE THE SOLUTION PLAN TO A PROGRAM

### FORTRAN

```
PROGRAM KILOTOMILES
C   Convert kilometers to miles
      INTEGER KILOMETERS
      REAL    MILES
C   Get kilometers from user
      PRINT *, 'Enter kilometers'
      READ *, KILOMETERS
C   Covert to miles
      MILES = KILOMETERS * 0.6241
C   Display miles to user
      PRINT *, 'Miles = ', MILES
      END
```