INDIAN INSTITUTE OF INFORMATION TECHNOLOGY

Dharwad
ज्ञानेन विकासः

Introduction to Programming and Data Structures

Dr. Animesh Chaturvedi

Assistant Professor: IIIT Dharwad

Young Researcher: Heidelberg Laureate Forum

Postdoc: King's College London & The Alan Turing Institute

PhD: IIT Indore MTech: IIITDM Jabalpur

HEIDELBERG LAUREATE FORUM

KING'S College LONDON

Indian Institute of Technology Indore
भारतीय प्रौद्योगिकी संस्थान इंदौर

PDPM
Indian Institute of Information Technology, Design and Manufacturing, Jabalpur

The Alan Turing Institute

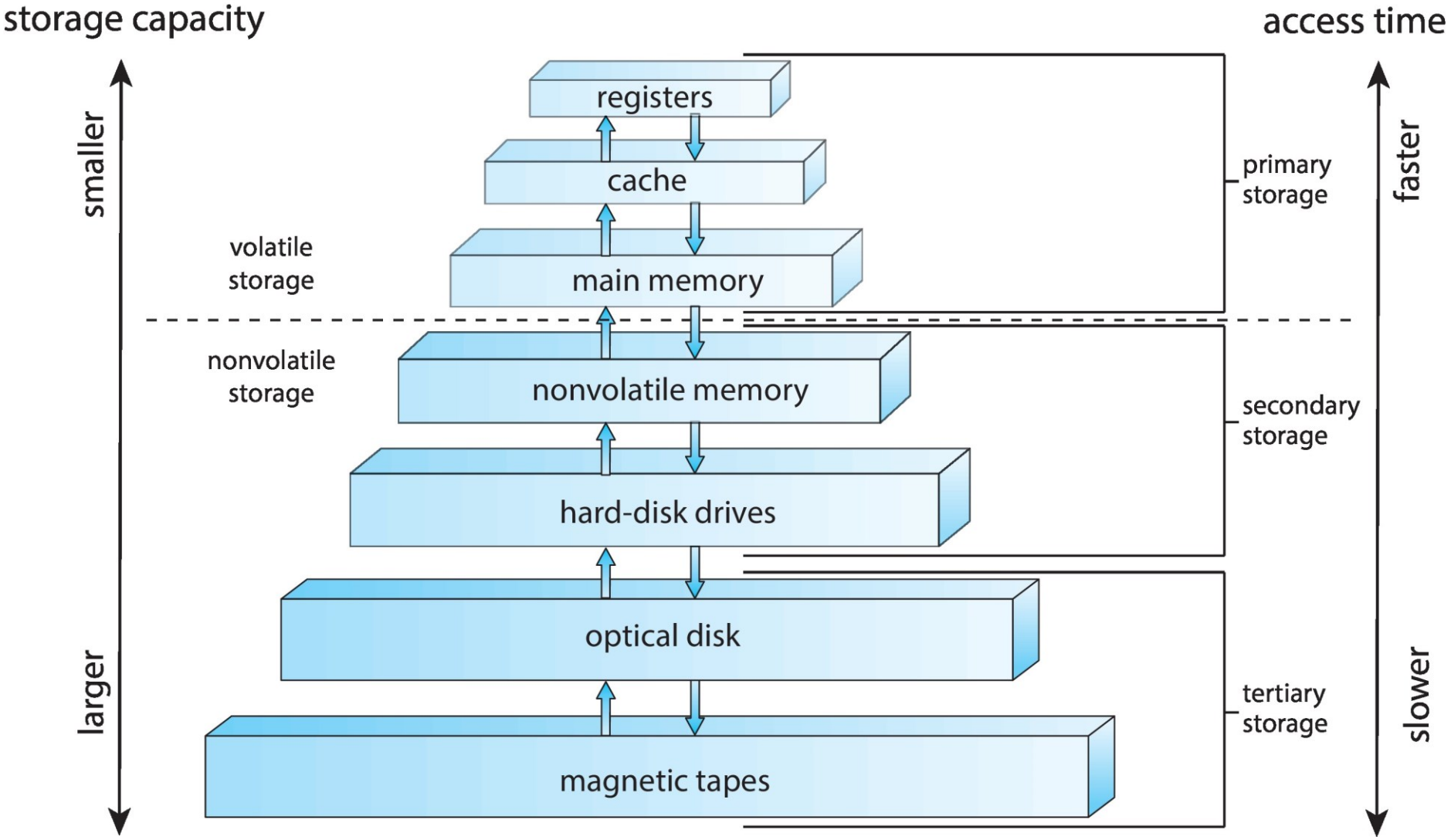
Goals of the Course

- To become familiar with Data structures and Algorithms
- To develop ability of problem solving skill in programming
- To understand that Data structures are the building blocks of larger softwares
- To develop ability for analyzing existing Data structures and Algorithms
- To develop skills with the C, C++, Java for Data structures and Algorithms

- "Get your data structures correct first, and the rest of the program will write itself."
  - David Jones

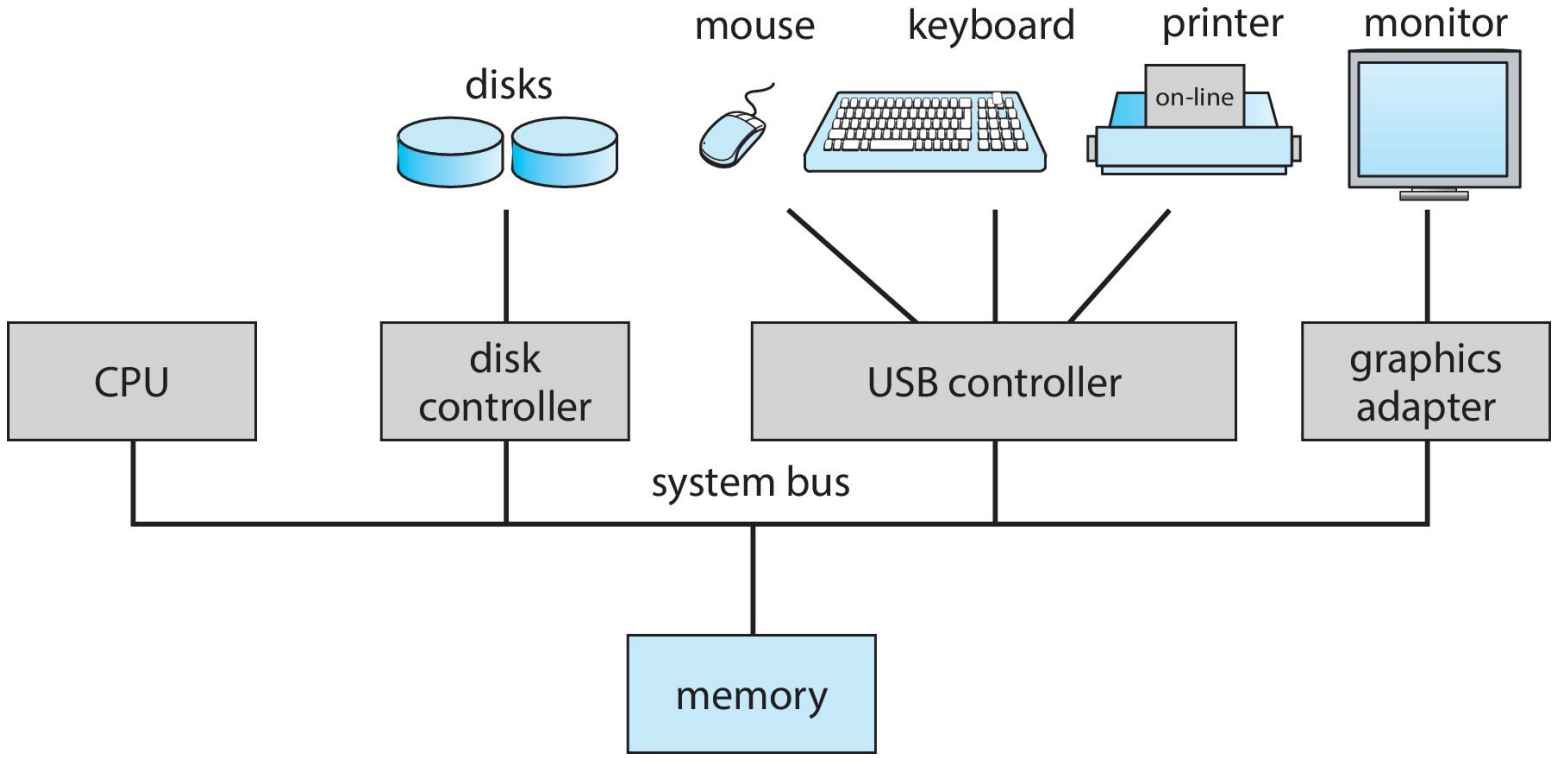
Basic Program Execution on CPU and Memory

Data Storage Device Hierarchy



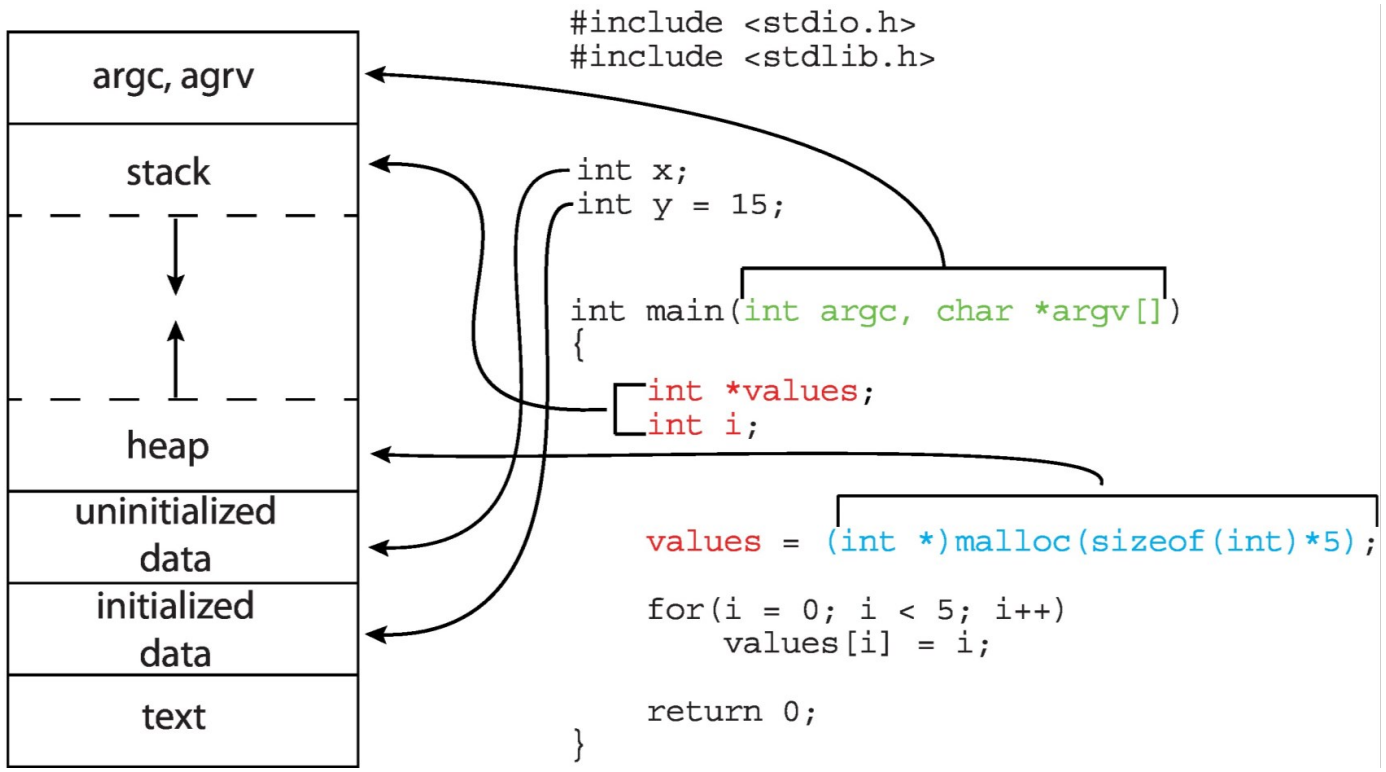
Computer System Organization

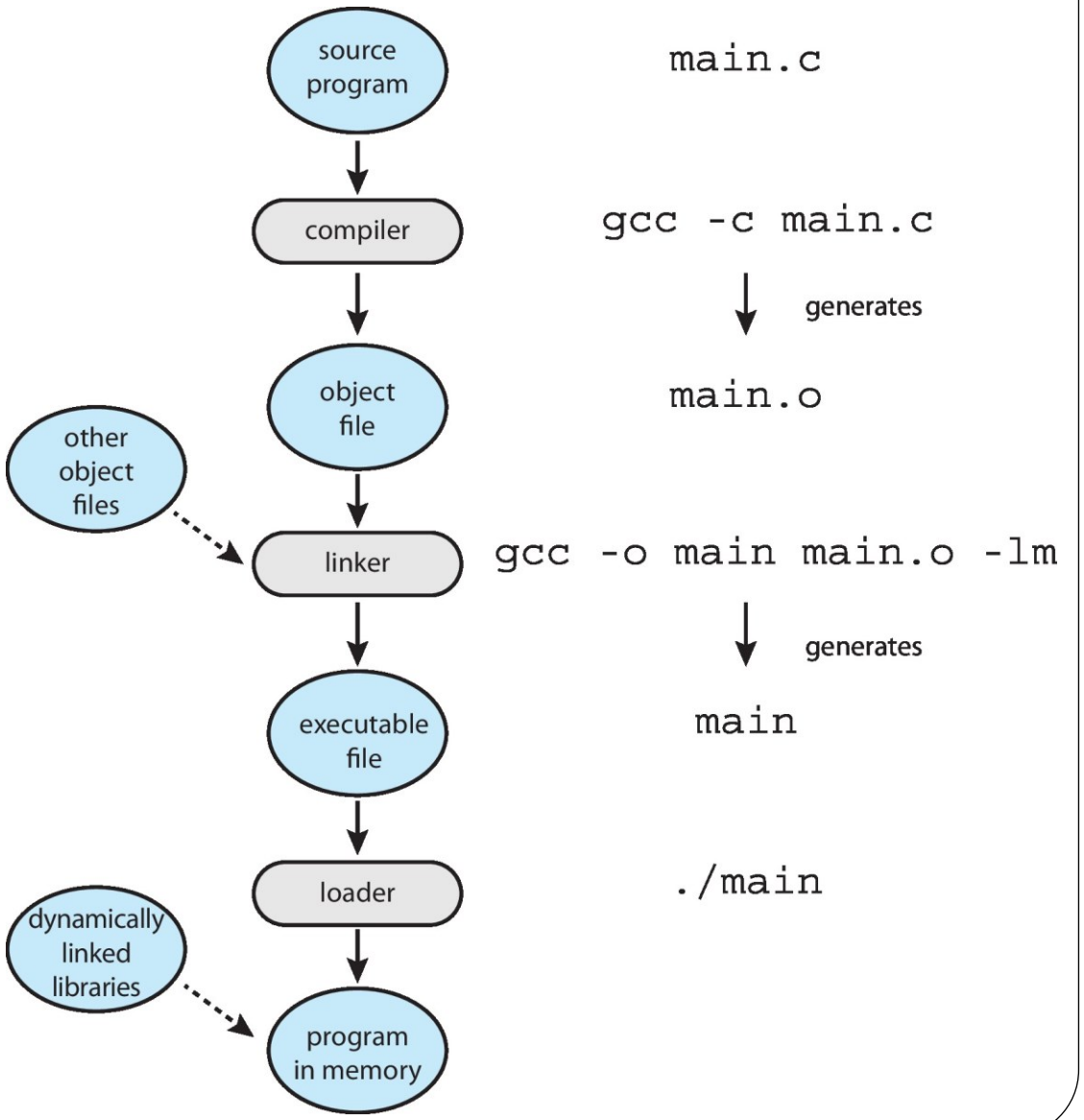
- Computer-system operation
  - One or more CPUs, device controllers connect through common **bus** providing access to shared memory
  - Concurrent execution of CPUs and devices competing for memory cycles



Program to Process

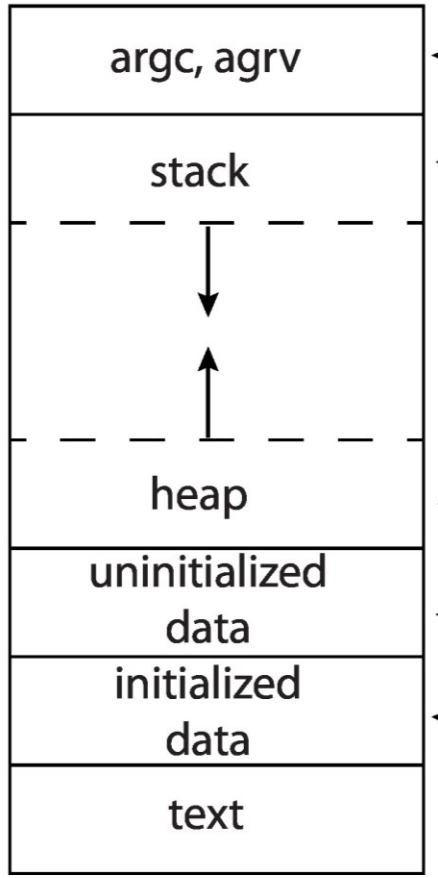
- When you run an exe file, the OS creates a process = a running program





Motivation: Program Code to Memory

- **Abstraction** of complex usage Program as a Memory (RAM or Cache).
- Conversion of **High level language to Low level language**
- Static/global variables are allocated in the executable
  - Local variables of a function on Stack
  - Dynamic allocation with malloc on the heap



Process as a Memory

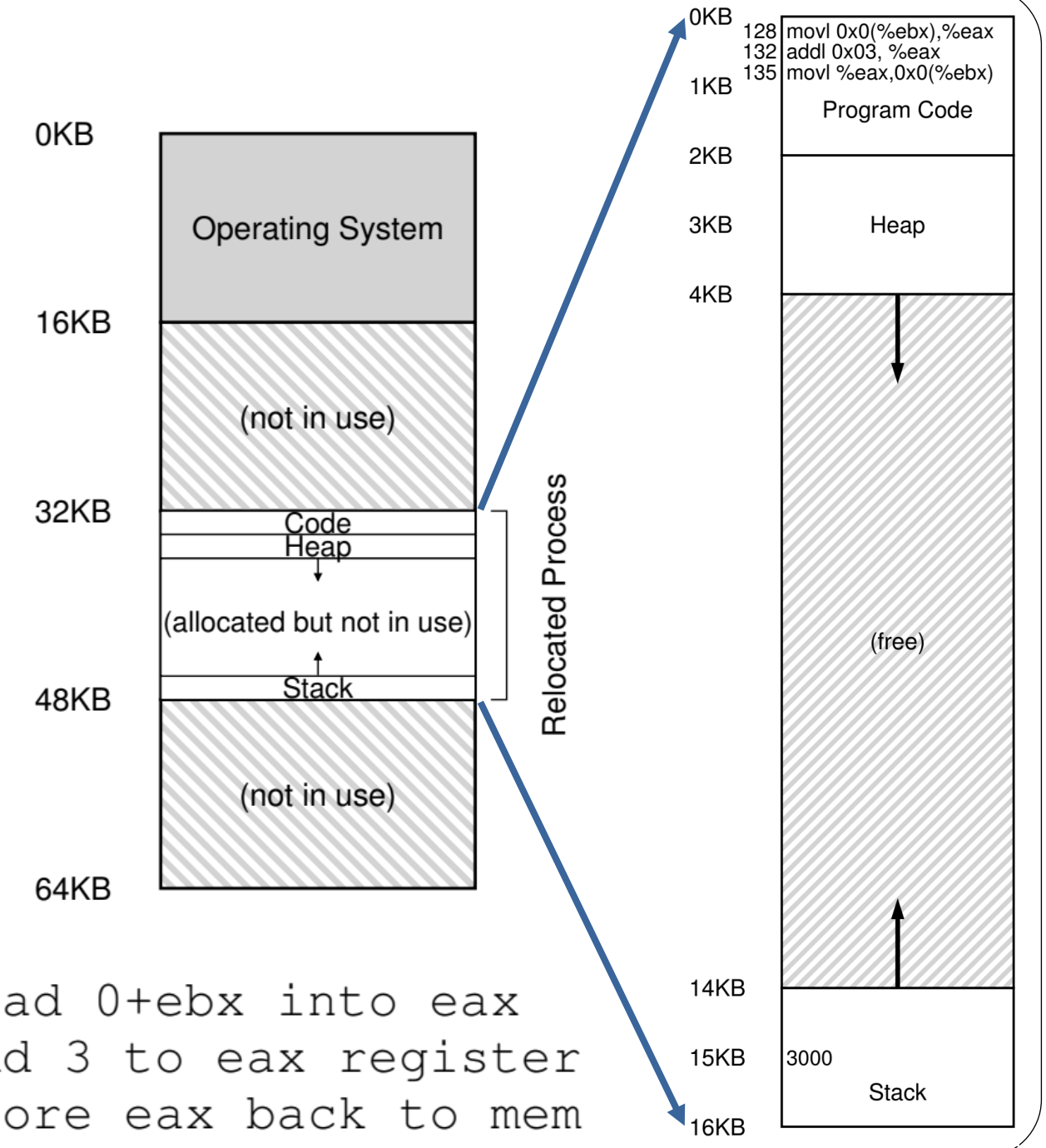
```
#include <stdio.h>
#include <stdlib.h>

int x;
int y = 15;

int main(int argc, char *argv[])
{
    int *values;
    int i;

    values = (int *)malloc(sizeof(int)*5);

    for(i = 0; i < 5; i++)
        values[i] = i;

    return 0;
}
```

Program as a Code

Program to Process

- Virtual address space is setup by OS during process creation

Simplified OS: places entire memory image in one chunk



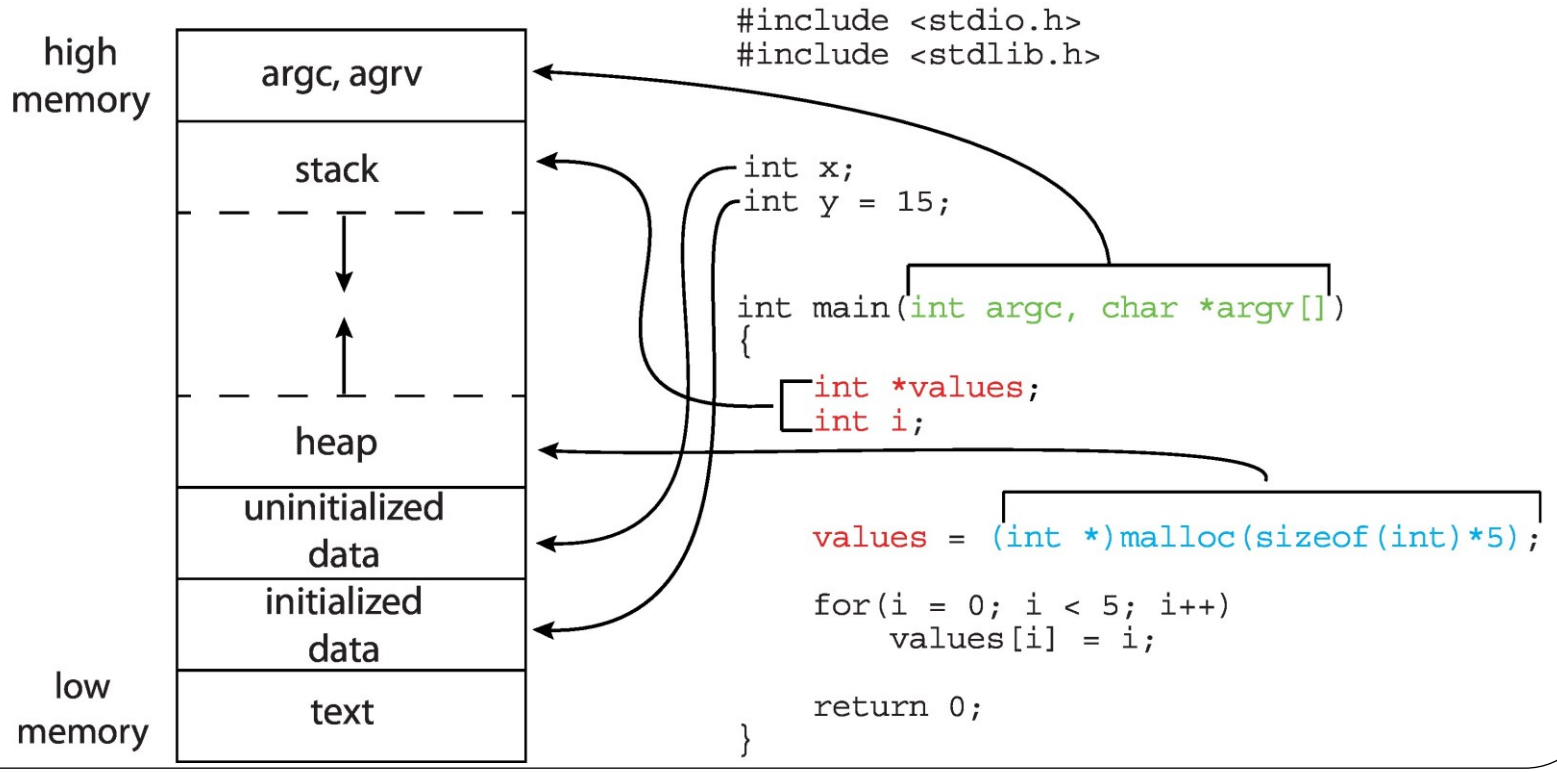

```
void func() {
    int x = 3000;
    x = x + 3;
    ...
```

Compiler

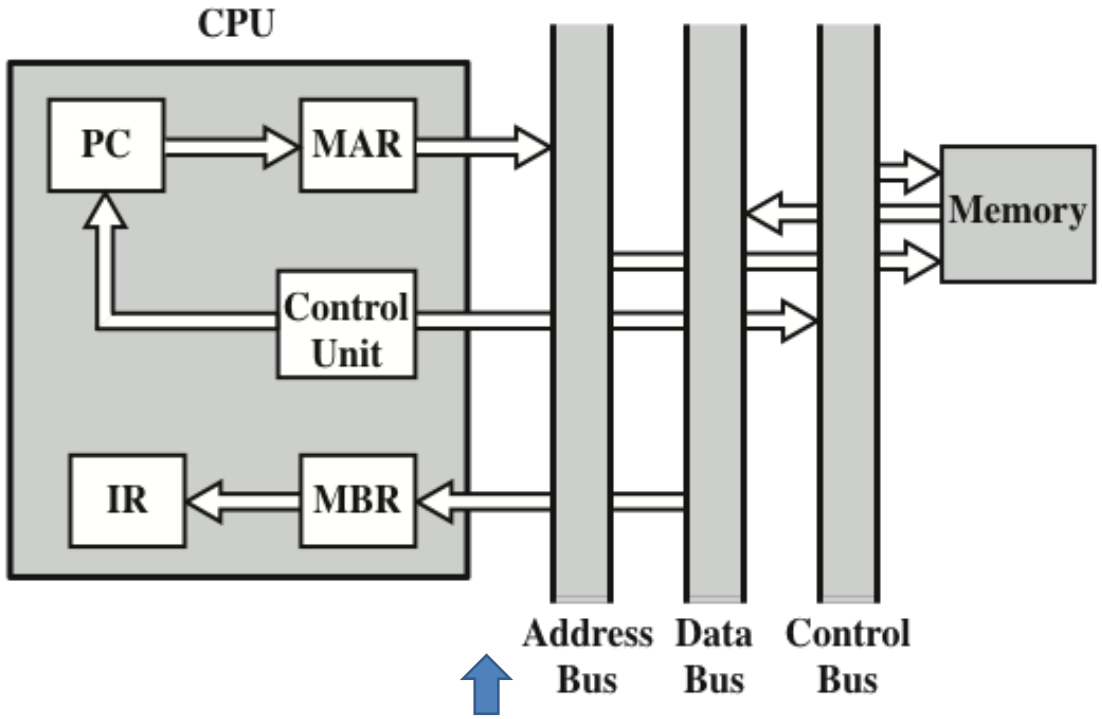
```
128: movl 0x0(%ebx), %eax     ;load 0+ebx into eax
132: addl $0x03, %eax         ;add 3 to eax register
135: movl %eax, 0x0(%ebx)     ;store eax back to mem
```

Program and Process

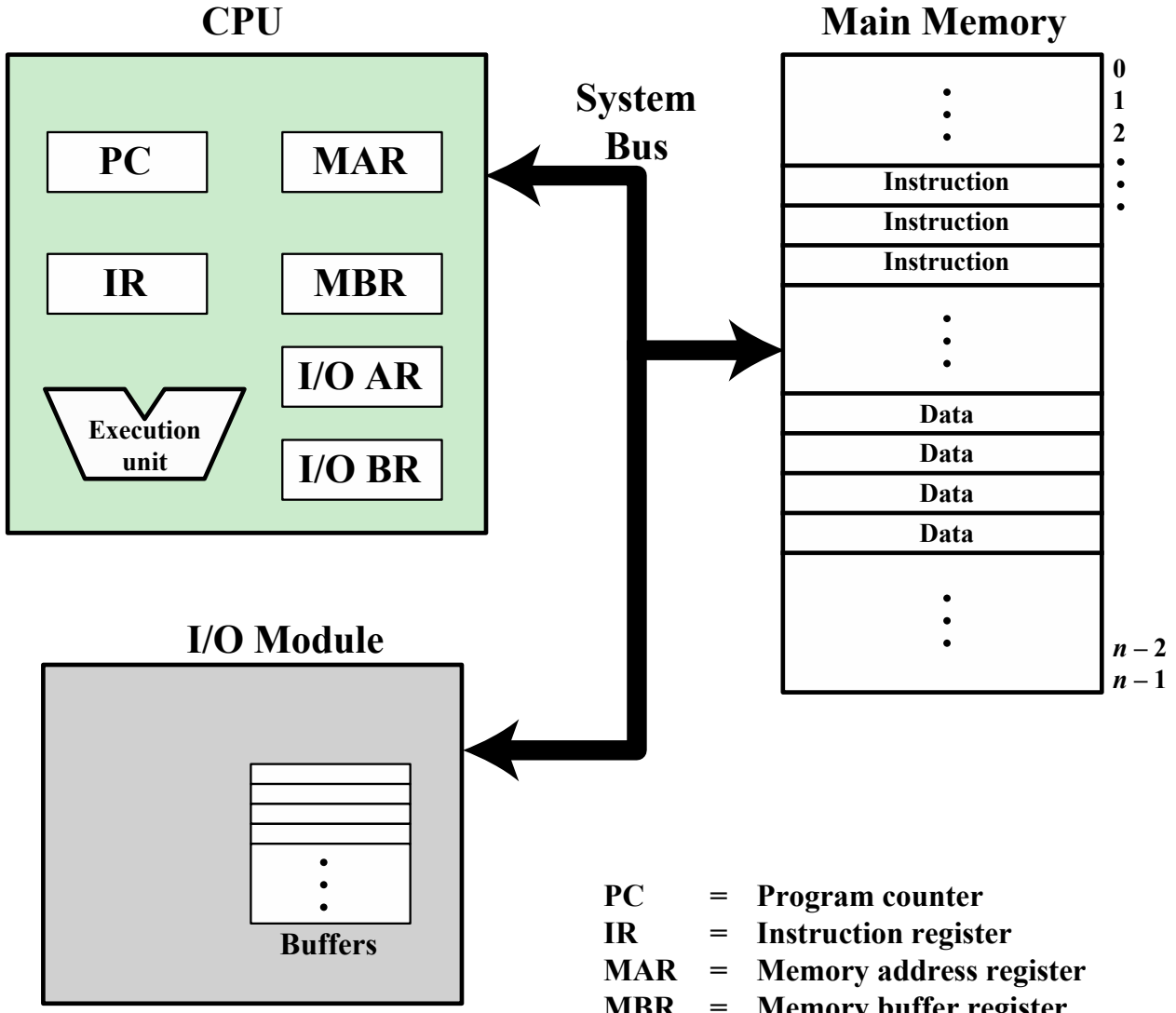
- A unique identifier
- Points CPU program counter to current instruction – Other registers may store operands, return values etc.
- CPU context: registers
  - Program counter
  - Current operands
  - Stack pointer



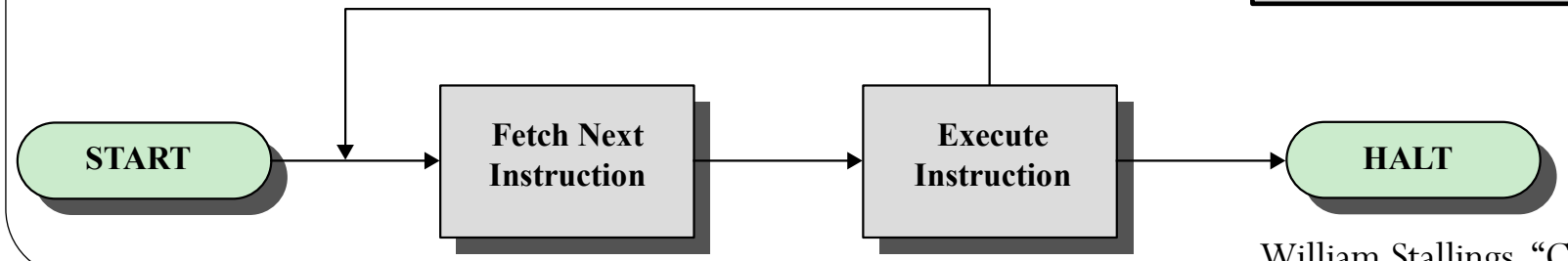
CPU and Memory



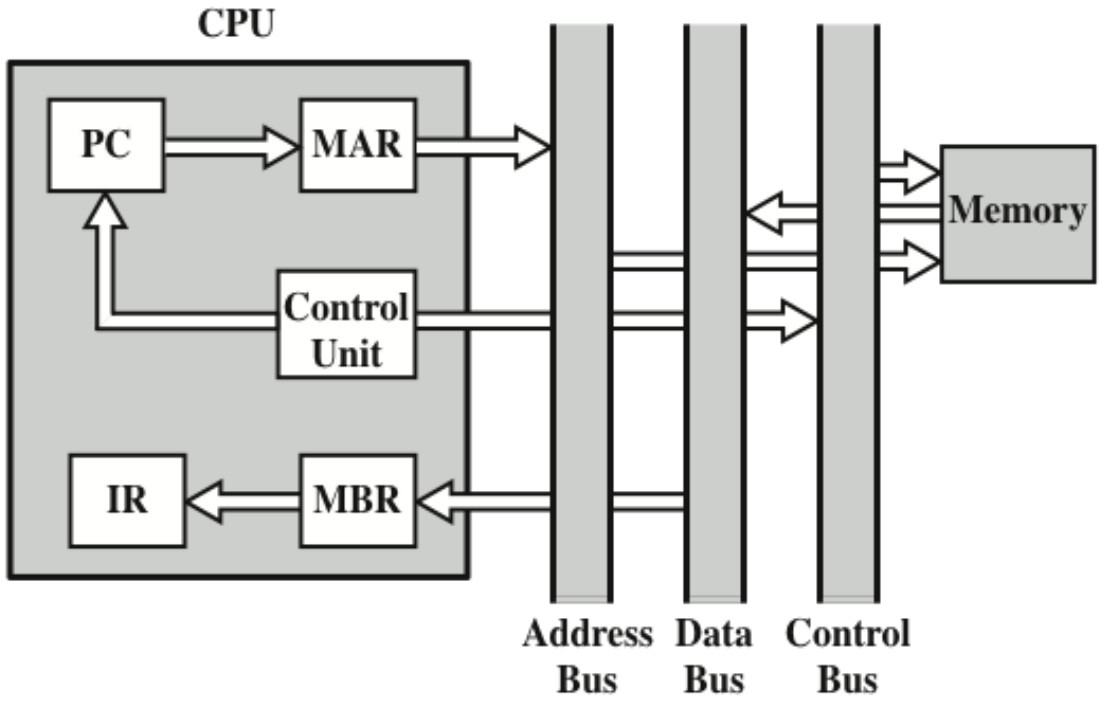
Fetch Cycle Execute Cycle



PC = Program counter
IR = Instruction register
MAR = Memory address register
MBR = Memory buffer register
I/O AR = Input/output address register
I/O BR = Input/output buffer register



CPU and Memory

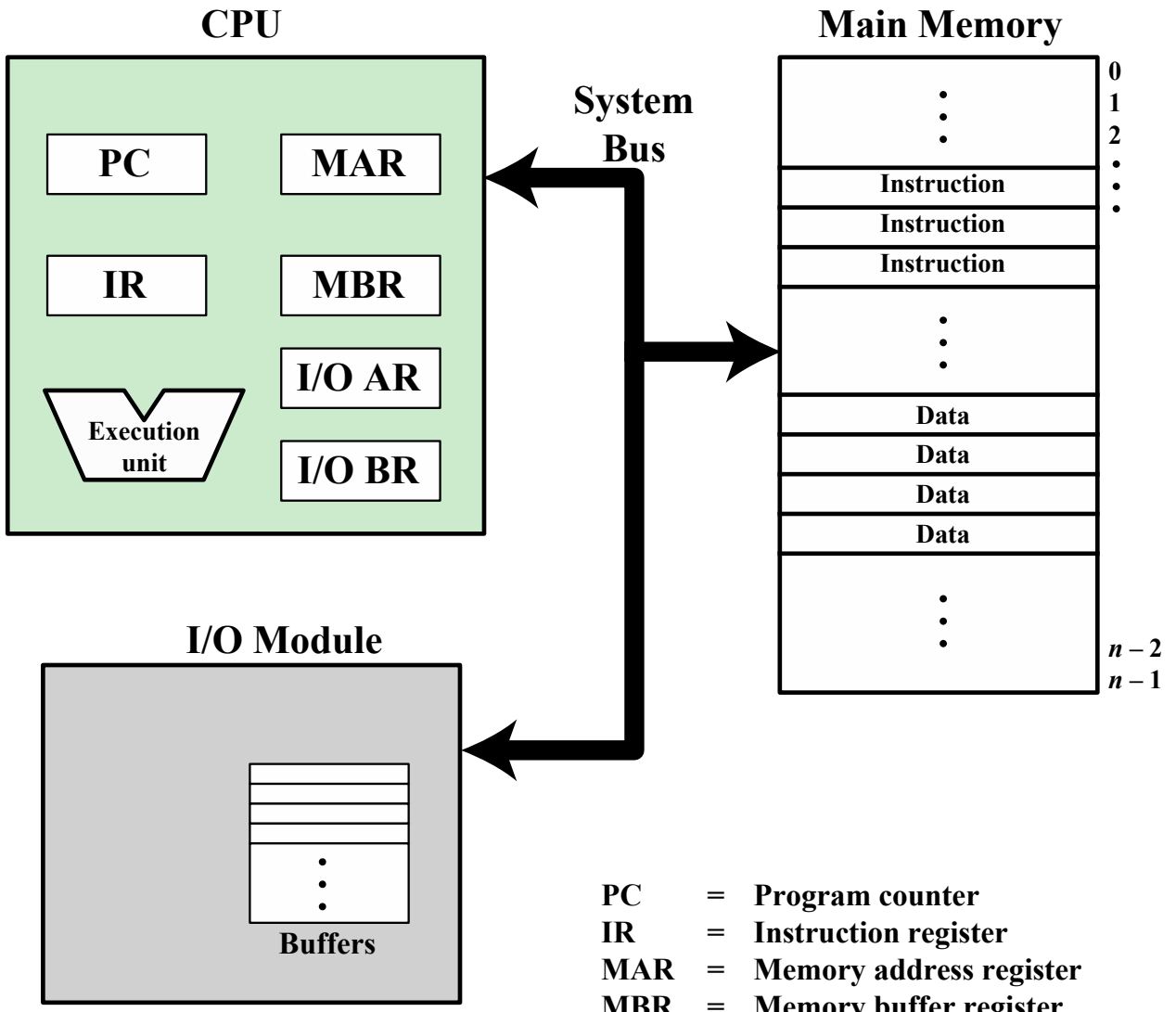


t1 : MAR ← PC

t2 : MBR ← MEMORY

PC ← (PC) + I

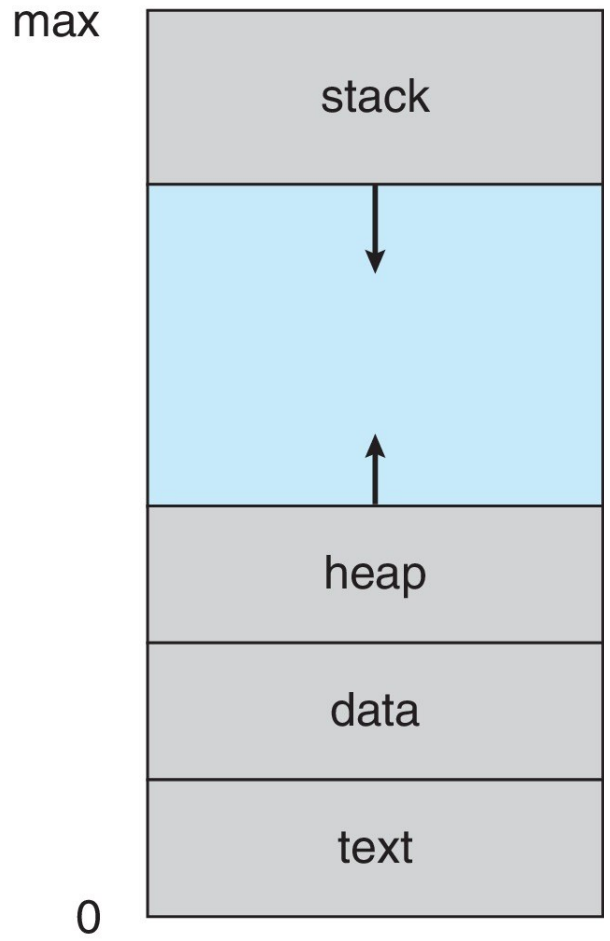
t3 : IR ← (MBR)



PC = Program counter
IR = Instruction register
MAR = Memory address register
MBR = Memory buffer register
I/O AR = Input/output address register
I/O BR = Input/output buffer register

Program and Process

- OS allocates memory and creates memory image
  - Loads code, data from disk exe
  - Creates runtime stack, heap
  - Opens basic files – STD IN, OUT, ERR
  - Initializes CPU registers – PC points to first instruction
- Memory image
  - Code & data (static)
  - Stack and heap (dynamic)



Basic Data Structures

Programing (Coding)

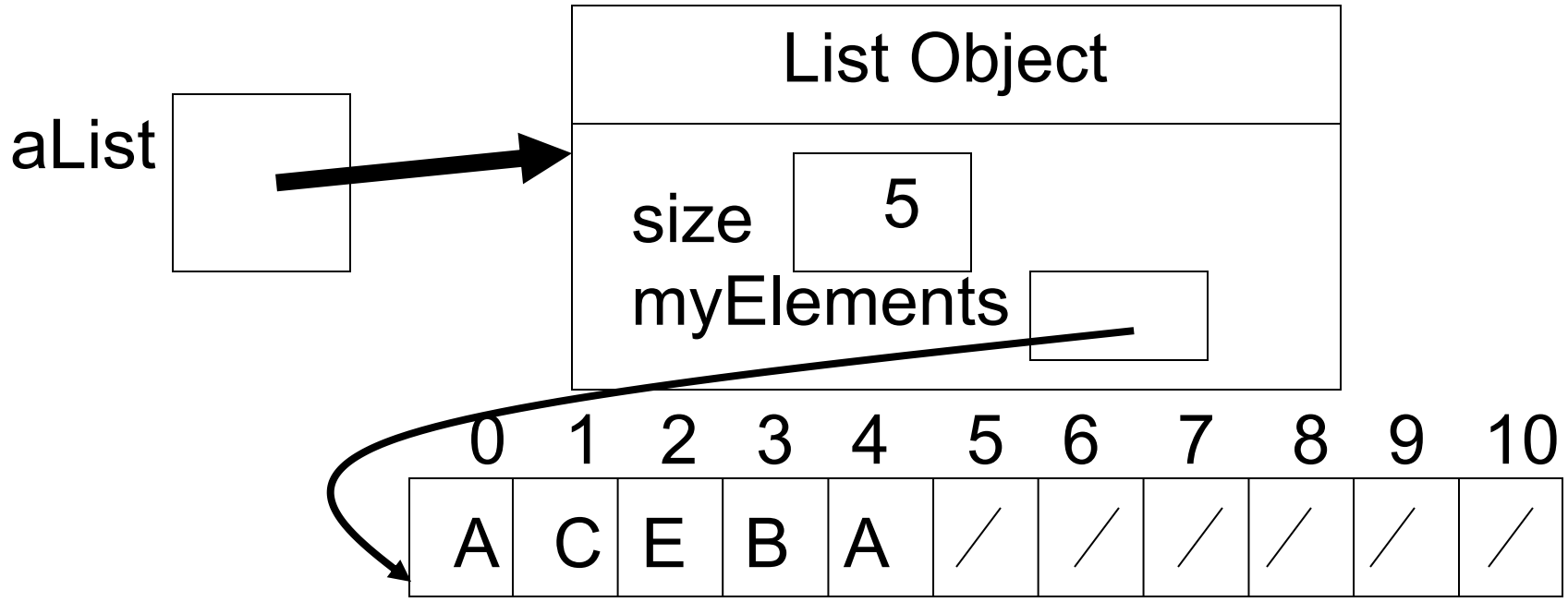
- All programs (or process) has data
  - To *store, display, gather*
  - In the form of *information, numbers, images, sound*
- Programmer decide to structure data
- Several option leads to program development
- Program properties
  - execution speed
  - memory requirements
  - maintenance (debugging, extending, etc.)

Code Implementation

- Theoretically
  - abstract base class describes ADT
  - inherited implementations implement data structures
  - can change data structures transparently (to client code)
- Practice
  - different implementations sometimes suggest different interfaces (generality *vs.* simplicity)
  - performance of a data structure may influence form of client code (time *vs.* space, one operation *vs.* another)

Data Structures

- A *Data Structure* is:
  - an implementation of an abstract data type *and*
  - "An organization of information, usually in computer memory", for better algorithm efficiency."



Data structures

Ideal data structure:

fast, elegant, memory efficient

Generates tensions:

- time *vs.* space
- performance *vs.* elegance
- generality *vs.* simplicity
- one operation's performance *vs.* another's

Dictionary ADT

- list
- binary search tree
- AVL tree
- Splay tree
- Red-Black tree
- hash table

Data structures

- Data Structures are containers:
  - they hold other data
  - arrays are a data structure
  - … so are lists
- Other types of data structures:
  - stack, queue, tree, binary search tree, hash table, dictionary or map, set, and on and on
  - www.nist.gov/dads/
  - en.wikipedia.org/wiki/List_of_data_structures
- Different types of data structures are optimized for certain types of operations

Data Structures Operations

- Data Structures will have 3 core operations
  - a way to add things
  - a way to remove things
  - a way to access things
- Details of these operations depend on the data structure
  - Example: List, add at the end, access by location, remove by location
- More operations added depending on what data structure is designed to do

Built-in Data Type

- There are five basic data types associated with variables
  - int - integer: a whole number
  - float - floating point value: a number with a fractional part
  - double - a double-precision floating point value
  - char - a single character
  - void - valueless special purpose type
  - int a = 4000; // positive integer data type
  - float b = 5.2324; // float data type
  - char c = 'Z'; // char data type
  - long d = 41657; // long positive integer data type
  - long e = -21556; // long -ve integer data type
  - int f = -185; // -ve integer data type
  - short g = 130; // short +ve integer data type
  - short h = -130; // short -ve integer data type
  - double i = 4.1234567890; // double float data type
  - float j = -3.55; // float data type

Built-in Data Type

Advantage

- Simple: Really simple! Only FIVE types of data!!
- Easy to handle: Allocation of memory and operations are already defined
- Built-in support by programming language
  - The C library are there to deal with them

Limitations

- There is a need for storing and handling variety of data types: Image, text, video, etc.
- Limited range
- Waste of memory
- No flexibility
- Error prone programming

User-Defined Data Types

- User can define their own data type
- Also, called Custom Data Type, Abstract Data Type (ADT), etc.
- All logically related data can be grouped into a form called structure
- Each member into the group may be a built-in data type or any other user defined data type
- No recursion, that is, a structure cannot include itself
- Examples:

Complex number: z = x + i y

Matrices: $A\_(m \times n)$

Date: dd/mm/yy

Date: {int dd, int mm, int yy}

User-Defined Data Types

Advantage

- It is always convenient for handling a group of logically related data items.
- Examples: Student's record, name, roll number, and marks.
- Elements in a set: Used in relational algebra, database, etc.
- A non non-trivial data structure becomes a trivial.
  - Helps in organizing complex data in a more meaningful way

Abstraction

- Because details of the implementation are hidden.
- When you do some operation on the list, say insert an element, you just call a function.
- Details of how the list is implemented or how the insert function is written is no longer required.

Abstract Data Type

- Algorithm: Description of a step-by-step process to solve a problem
  - independent of High Level Language (HLL)
- Data Structure
  - A set of algorithms which implement an Abstract Data Type (ADT)
  - Data Structures: Arrays, Linked lists, Stacks, Queues, Matrices, Trees, Graphs
  - Usage: Searching, Sorting
- Abstract Data Type
  - An opportunity for an acronym
  - Mathematical description of an object and the set of operations on the object

Abstract Data Type

- Present an ADT
- Motivate with some applications
- Repeat until browned entirely through
  - develop a data structure for the ADT
  - analyze its properties
    - efficiency
    - correctness
    - limitations
    - ease of programming
- Contrast data structure's strengths and weaknesses
  - understand when to use each one

Abstract Data Types

- Abstract Data Types (aka ADTs) are descriptions of how a data type will work without implementation details
- Description can be a formal, mathematical description
- Java interfaces are a form of ADTs
  - some implementation details start to creep in

Abstract Data Types

- Programming languages usually have a library of data structures
  - Java collections framework
  - C++ standard template library
  - .Net framework (small portion of VERY large library)
  - Python lists and tuples
  - Lisp lists

References

Images and Text from basics of

- CSE326: Data Structure, Department of Computer Science and Engineering, University of Washington https://courses.cs.washington.edu/courses/cse326
- Mike Scott, CS 307 Fundamentals of Computer Science, https://www.cs.utexas.edu/~scottm/cs307/
- William Stallings. "Computer Organization and Architecture". 10th Edition
- Mythili Vutukur. Lectures on Operating Systems, Department of Computer Science and Engineering, IIT Bombay, https://www.cse.iitb.ac.in/~mythili/os/
- Avi Silberschatz, Peter Baer Galvin, and Greg Gagne. Operating System Concepts (Tenth Edition). https://www.os-book.com/OS10/slide-dir/index.html
- Online textbook Operating Systems: Three Easy Pieces (OSTEP)

ขอบคุณ
Thai

Grazie
Italian

תודה רבה
Hebrew

धन्यवादः
Sanskrit

ಧನ್ಯವಾದಗಳು
Kannada

Ευχαριστώ
Greek

Thank You
English

Gracias
Spanish

Спасибо
Russian

Obrigado
Portuguese

شكراً
Arabic

https://sites.google.com/site/animeshchaturvedi07

Merci
French

多謝
Traditional
Chinese

धन्यवाद
Hindi

Danke
German

多谢
Simplified
Chinese

நன்றி
Tamil
Tamil

ありがとうございました
Japanese

감사합니다
Korean