# Operating Systems

## Lecture #1: Basic concepts of O/S

Written by David Goodwin
based on the lecture series of Dr. Dayou Li
and the book *Understanding Operating Systems* $4^{th}$ *ed.*
by *I.M.Flynn and A.McIver McHoes* (2006)

Department of Computer Science and Technology,
University of Bedfordshire.

Operating Systems, 2013

$28^{th}$ January 2013

# Outline



David Goodwin
University of Bedfordshire





# Lectures

# Lecturing schedule



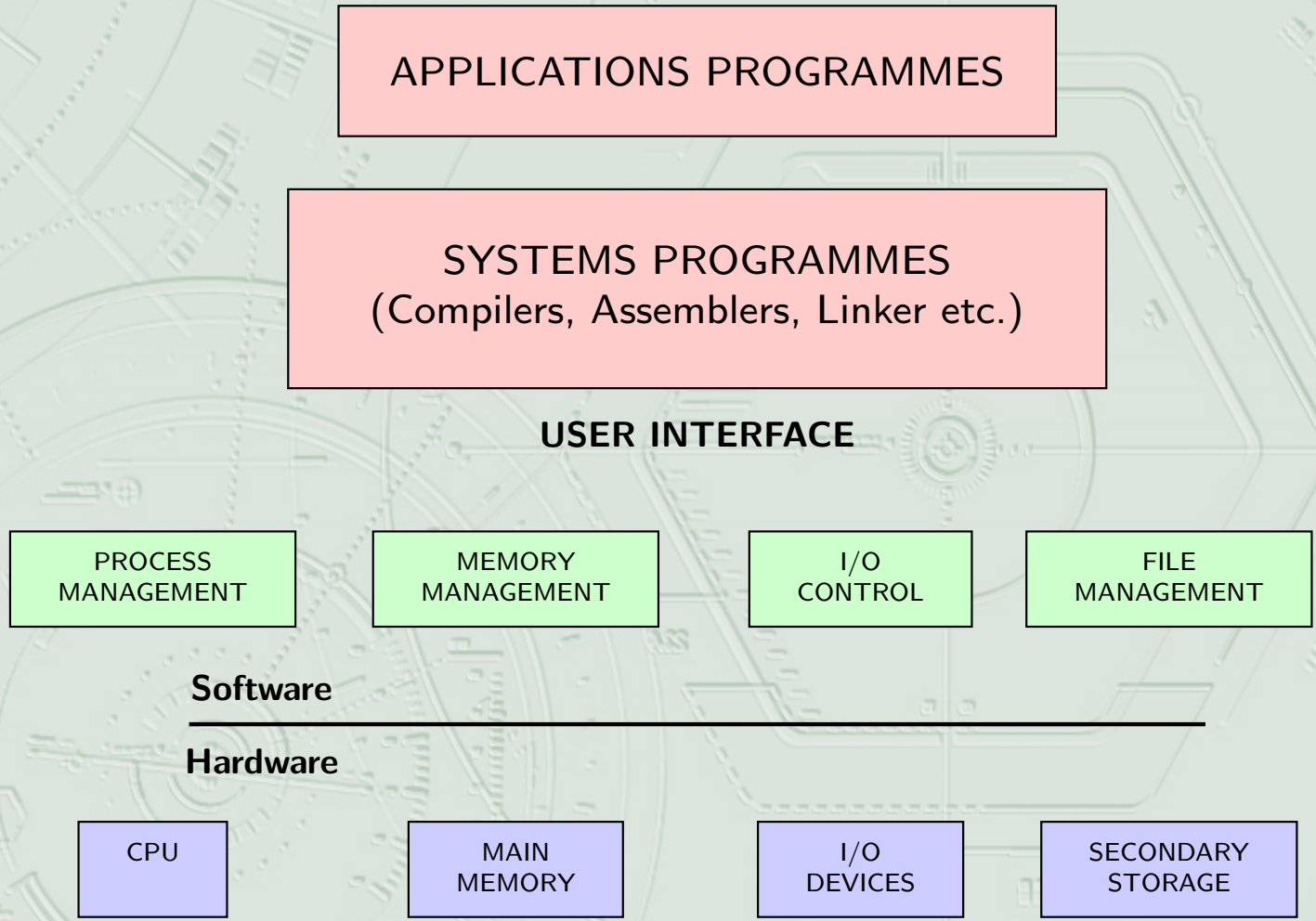
1. Operating system concepts
2. Early paradigms of memory management
3. Modern memory management techniques
4. I/O device management
5. File management
6. Process management
7. Deadlock resolution
8. Critical sections
9. Concurrent processes
10. Security and ethics
11. Network organisation and management
12. *Case Study*: Memory management in real O/S
13. *Case Study*: Process management in real O/S
14. Revision



# Computer systems

# O/S in computer systems





# O/S in computer systems



**Hardware** is the term used to describe all the physical electronic and mechanical elements forming part of a computer system. Provides basic computing resources (CPU, memory, I/O devices)

**Software** is the term used to describe the instructions or programs that the hardware needs in order to function.

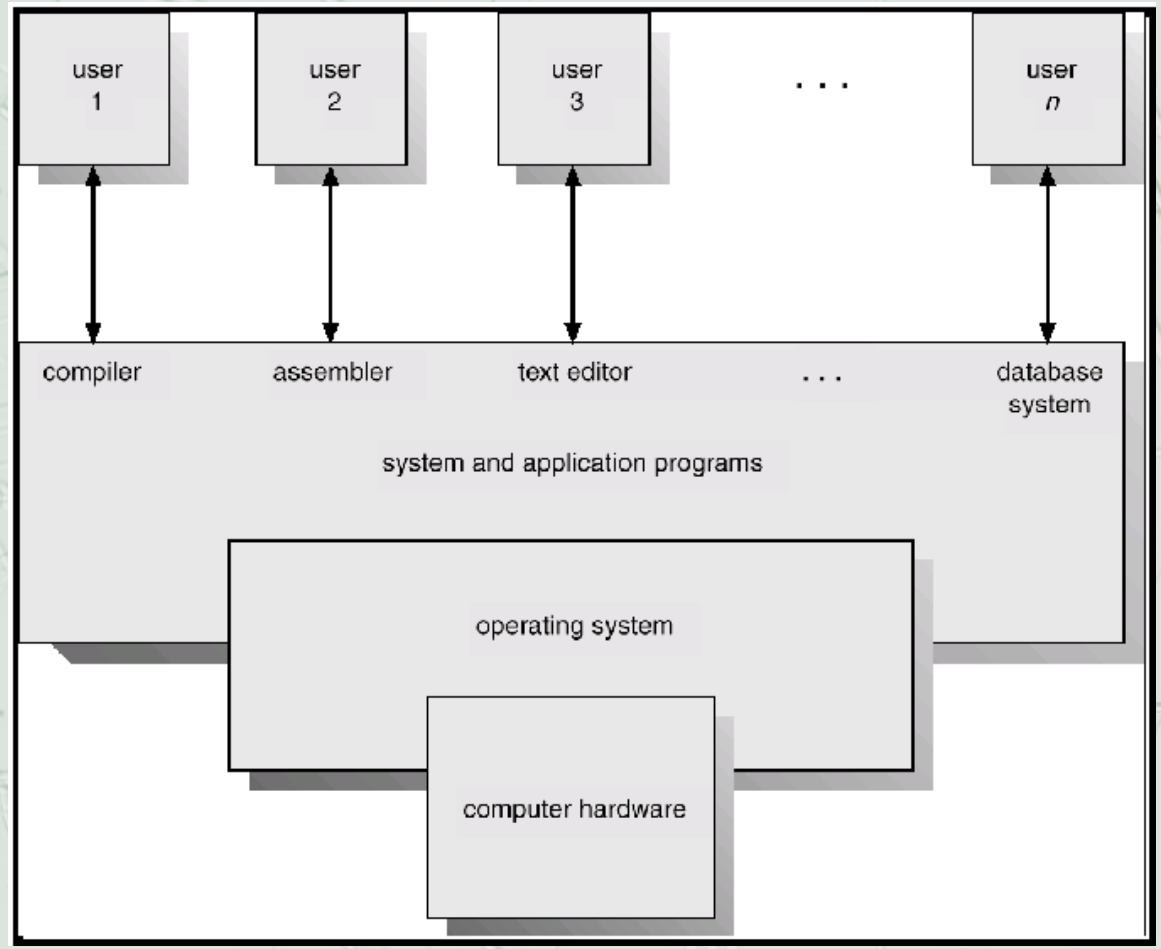
**Operating system** controls and coordinates the use of the hardware among the various application programmes for various users

**Applications programmes** define the ways in which the system resources are used to solve the computing problems of the users (compilers, database systems, video games, business programmes, etc.)

**Users** (people, machines, other computers)

# O/S in computer systems





# O/S in computer systems



**Resource allocator** manages and allocates resources

**Control programme** controls the execution of user programs and operations of I/O devices

**Kernel** the one programme running at all times (all else being application programmes)



# Concepts of O/S

# Basic concepts of O/S



- O/S definition
  - O/S is a piece of software that controls every file, every device, every section of main memory and every nanosecond processing time

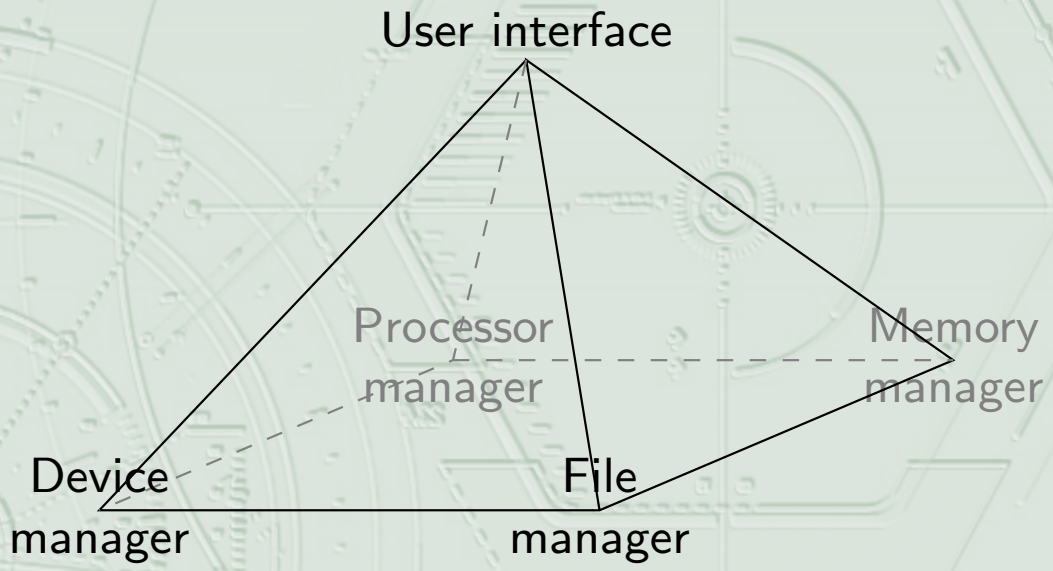



Figure : Model of an operating system showing four sub-system managers supporting the user interface

# Basic concepts of O/S



**Memory manager** is in charge of main memory (RAM) in terms of checking validity of each request for memory space

**Processor manager** keeps track of status of each process

**Device manager** monitors devices channels and contorl units and decides the most efficient way to allocate the devices

**File manager** keeps track of every file



# Types of O/S

# Types of O/S



- O/S classification
  - Single user
  - Multiple tasks
  - Batch systems
  - Real-time systems
  - Hybrid systems
  - Embedded systems

# Single user O/S



- Provides capability to perform tasks on the computer system such as writing programmes and documents, priniting and accessing files
- Provides access to the computer system by a single user at a time, e.g. typical home computers
- Most Windows are single user operating systems

# Multi-task O/S

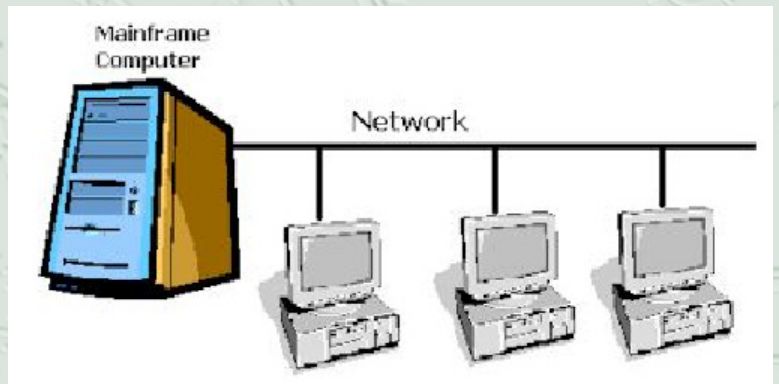


- A mutli-tasking operating system provides the ability to run more than one programme at a time, e.g. word processing, printing a document, copying files to a flash memory stick. Each of the tasks the user is performing appears to be running at the same time.
- A multi-tasking operating system has the **advantage** of letting the user run more than one task at a time so as to increase productivity.
- The **disadvantage** is that the more programmes that are run by the user, the more memory that is required.

# Multi-task O/S



- Manage and run all user requests, ensuring they do not interfere with each other, e.g. printer queue
- Allow more than one user to access the computer system at a time, normally via a network, e.g. Unix
- Much more complex single-user operating system



# O/S utilities



- To manage the computer system, users, printers, files and write programmes, the operating system is generally provided with a number of utility programmes. The utilities are used for:
  - Managing files and documents
  - Development of programmes and software
  - Communicating between people and with other computer systems
  - Managing user requirements for programmes, storage space and priority

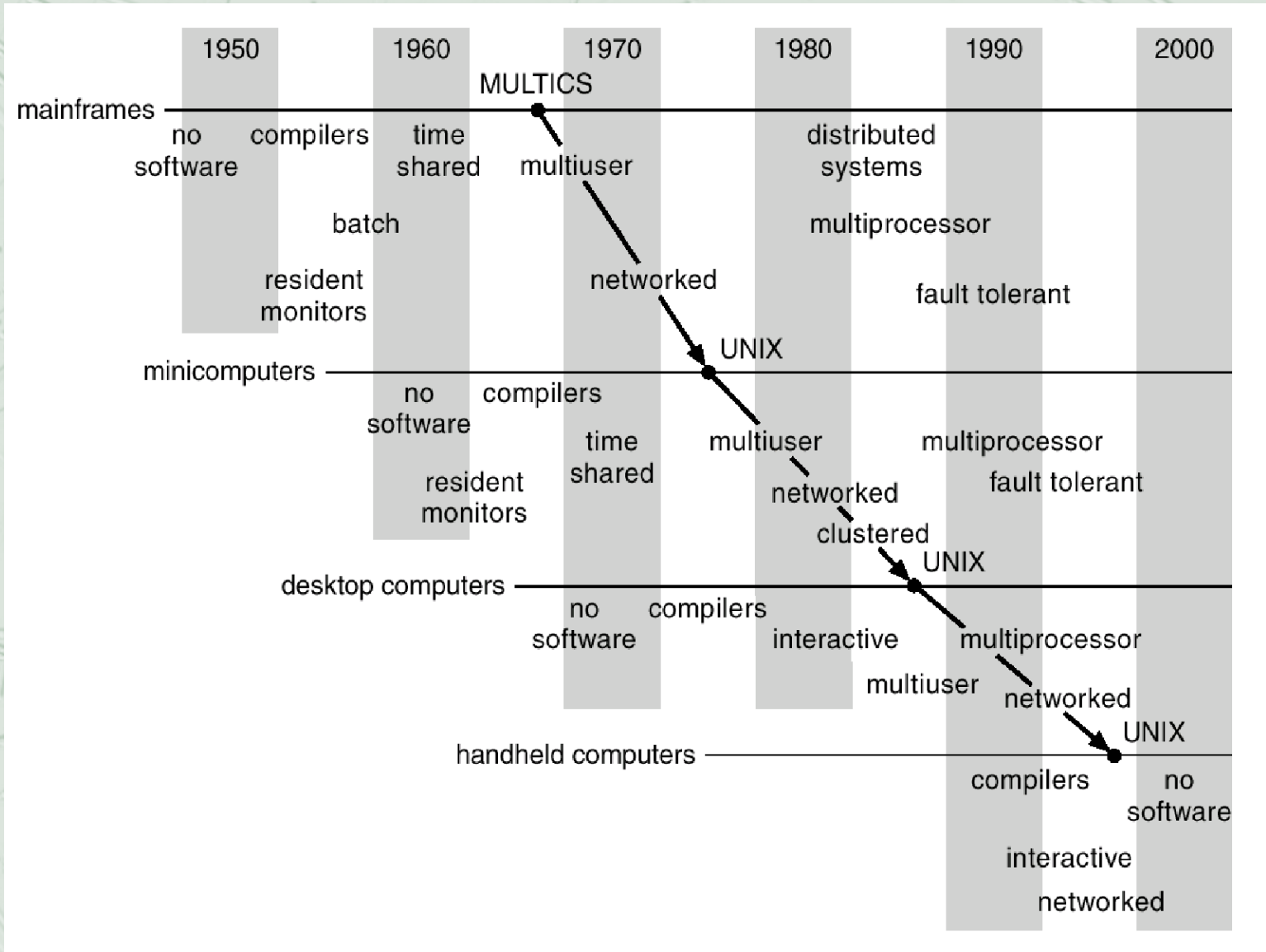
# O/S utilities



- The operating system provides each user with an interface that accepts, interprets and executes user commands or programmes
- This interface is commonly called a SHELL or Command Line Interpreter (CLI)
- In some systems this might be a simple text mode line-by-line entry using keywords (such as DOS or UNIX)
- In other systems it might be highly graphical using windows and pointing device such as a mouse (such as Windows)

# Migration of O/S features

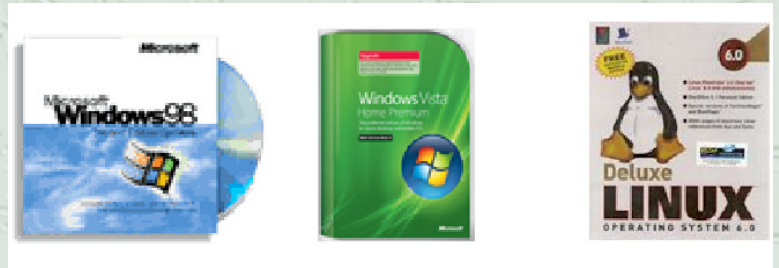




# Some O/S products



- Typical operating systems include:
  - Unix/Linux
  - Windows 3.x/9x/NT/2000/XP/Vista/7
  - Mac OS
  - Windows Mobile
  - Symbian
  - DOS
  - VMS





# CPU structure

# CPU structure

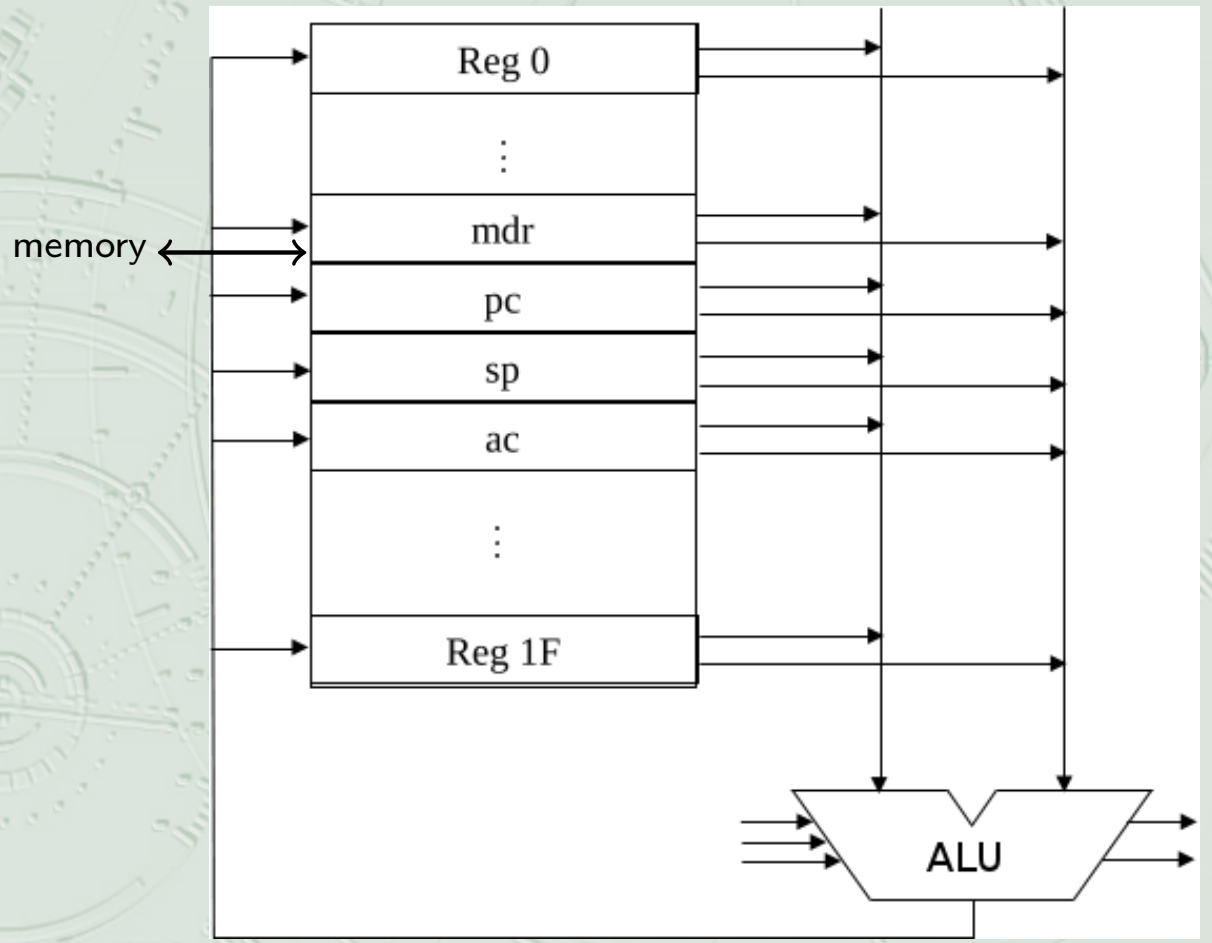


- A CPU consists of two sections
  1. Computational section
  2. Control section

# CPU structure



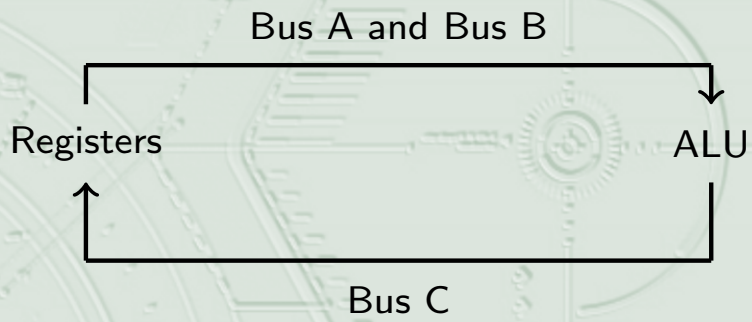
- Computational section
  - Structure



# CPU structure



- ALU (arithmetic logic unit)
  - Performing arithmetic, shifting, logical AND, complementation, bypass operations
  - Datapath:

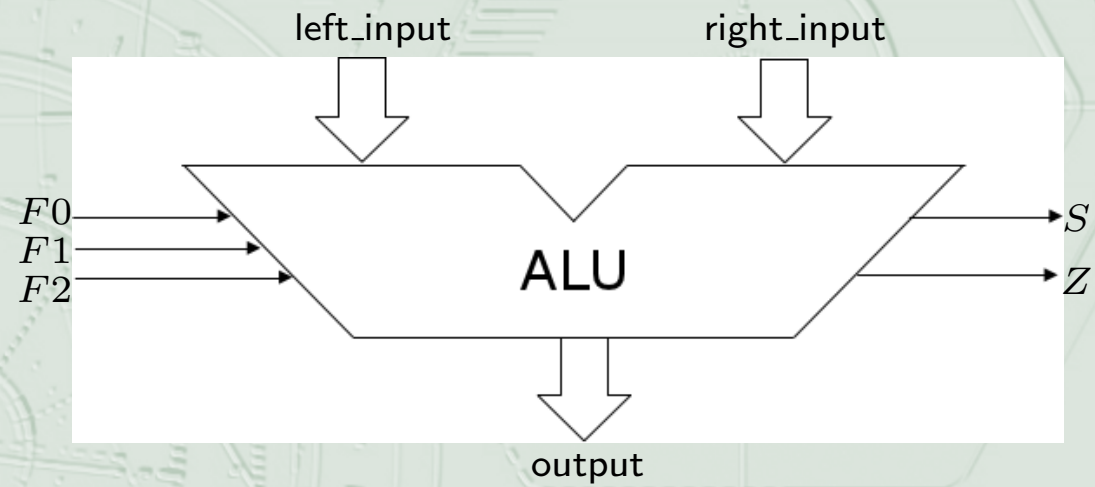



  - Control "tell" the ALU which operation to perform
  - $S$: the sign bit of the output
  - $Z$: bitwise NOT-OR of output (i.e. $Z = 1$ when all output bits are $0$)

# CPU structure



- ALU is the heart of a CPU
  - It is where computation carries out
  - It performs shift and addition (consists of shifters and adders)
  - It performs arithmetic, logical and shifting operations



# CPU structure



| $F0$ | $F1$ | $F2$ | Output |
|---|---|---|---|
| 0 | 0 | 0 | left_input |
| 0 | 0 | 1 | bitwise complement of left_input |
| 0 | 1 | 0 | AND |
| 0 | 1 | 1 | multiplication |
| 1 | 0 | 0 | addition |
| 1 | 0 | 1 | subtraction |
| 1 | 1 | 0 | left shift left_input one position |
| 1 | 1 | 1 | right shift left_input one position |

# CPU structure



- Tasks and registers
  - *mask* to obtain specific part of data
  - *read* from or *write* to memory
  - *keep track* of current position of the execution of a programme
  - temporarily store an instruction of a programme
  - store intermedium results
  - store stack position

# CPU structure



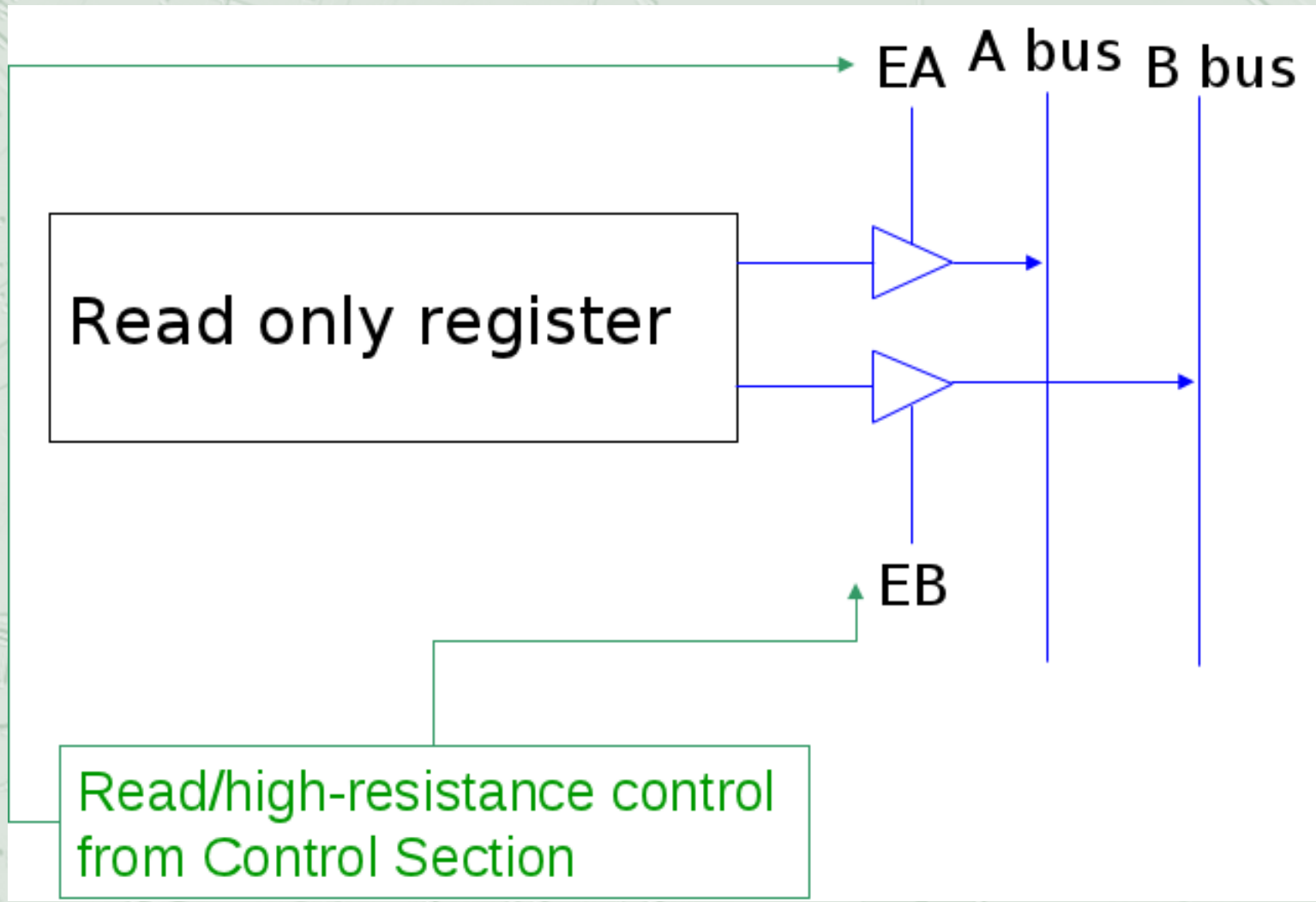
- Mask registers
  - Registers 0 to 4 contain constants for the purpose of mask

| Reg | contents | name |
|---|---|---|
| 0 | 0000 | |
| 1 | 0001 | |
| 2 | 0FFF | Xmask |
| 3 | 00FF | Ymask |
| 4 | 000F | Zmask |

  - Mask is performed by using bitwise AND
  - They are read-only registers

# CPU structure

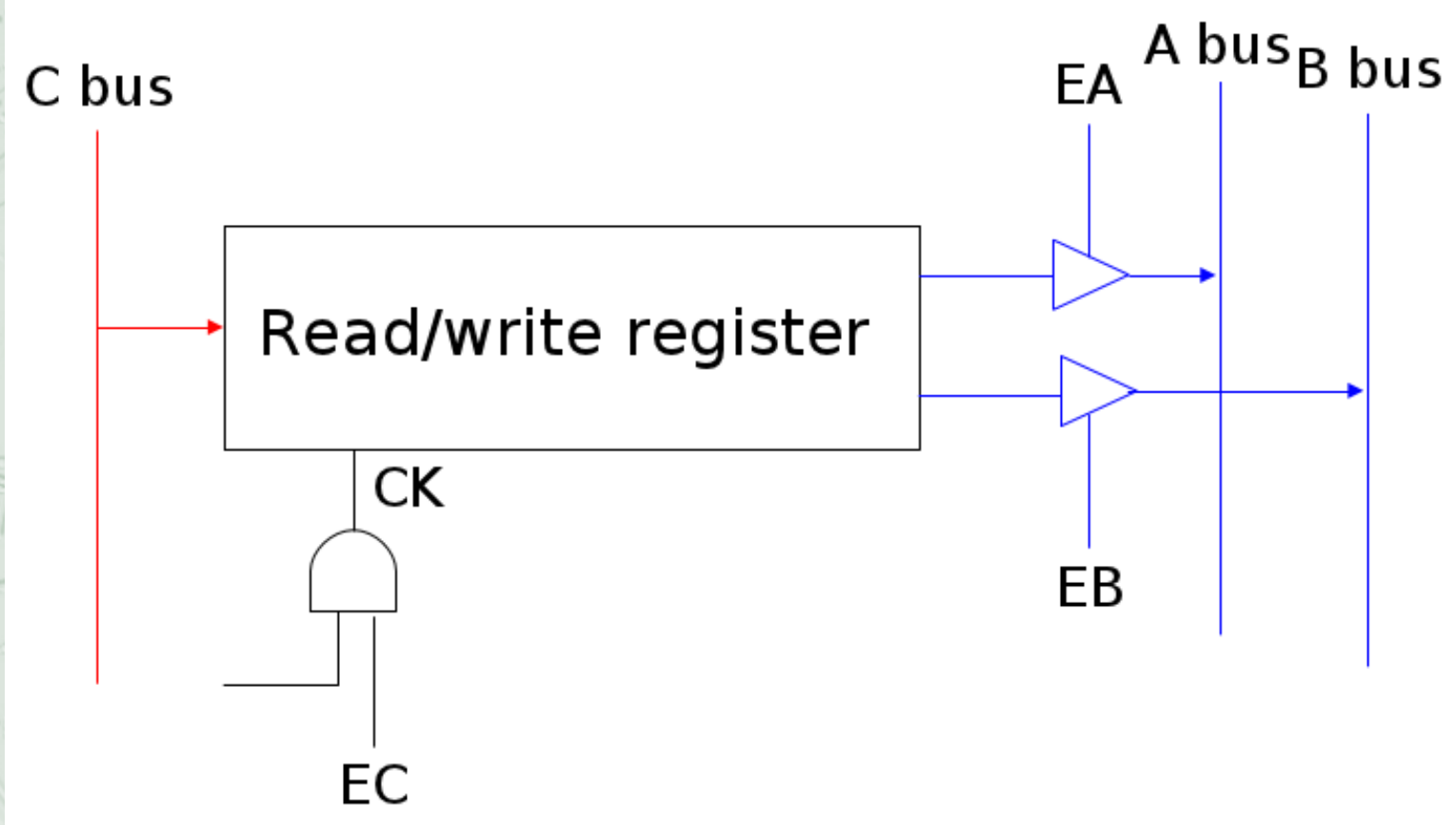




# CPU structure



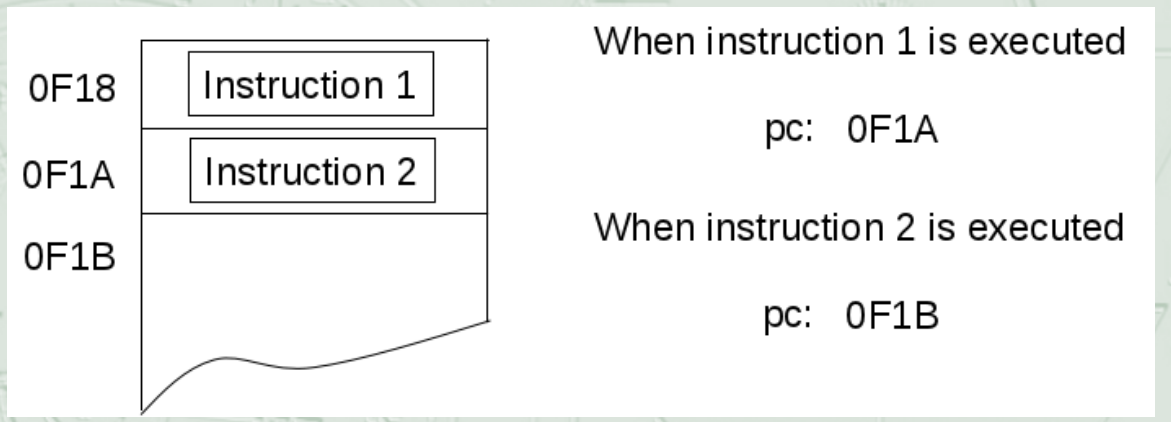
- Accumulation register (ac)
  - register-to-register operations save intermedium results to this general purpose register
  - read and write register



# CPU structure



- Program counter register (pc)
  - Storing the memory address of the next instruction of a program so that it functions like a pointer (or a bookmark)
  - Read and write register



# CPU structure



- Stack pointer register (sp)
  - local variables and parameters are stored in the stack
  - sp gives the current position of stack
  - read and write register

# CPU structure

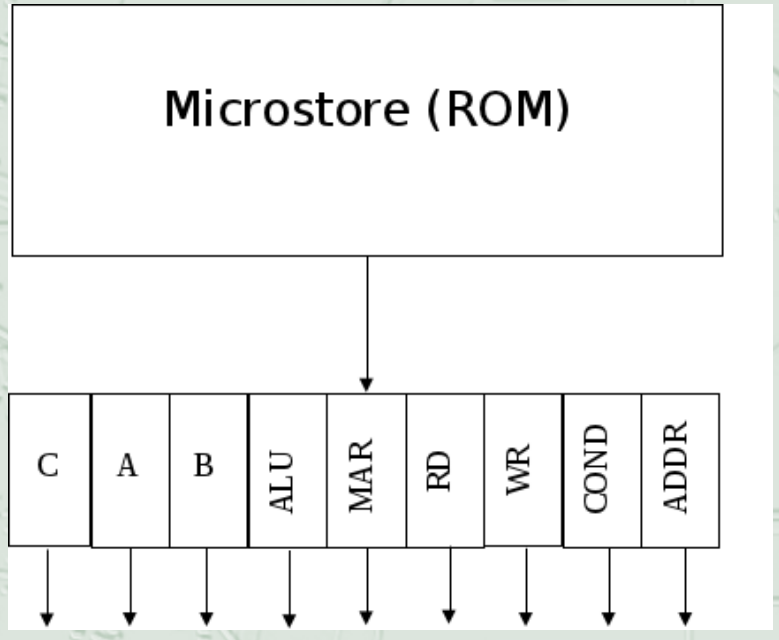


- Memory data register (mdr)
  - CPU reads data from memory and temporarily saves the data to this register (it is also known as memory buffer register)
  - When CPU writes data to memory, it also temporarily saves data in this register
  - read and write register
- Memory address register (mar)
  - storing memory address where CPU will either read data from the addressed memory cell or write data to the memory cell
  - read and write register

# CPU structure



- Control Section
  - Structure



# CPU structure



- Controls
  - A B and C generate EA, EB, and EC signals, respectively
  - ALU generates F0, F1, and F2 control signals for ALU
  - MAR generates EC signal for mar
  - RD generates read signal to set memory to such a state the CPU can read data from the memory
  - WR generates write signal to set memory to such a state that CPU can write data to the memory
  - COND generates two control signals for a so-called branch multiplexer which has 4 inputs: 0, 1, Z, and S
  - Mircostore is a ROM that consists of a number of micro-instructions, each of which has an address field called ADDR