Introduction

Chapter 1

S. Dandamudi

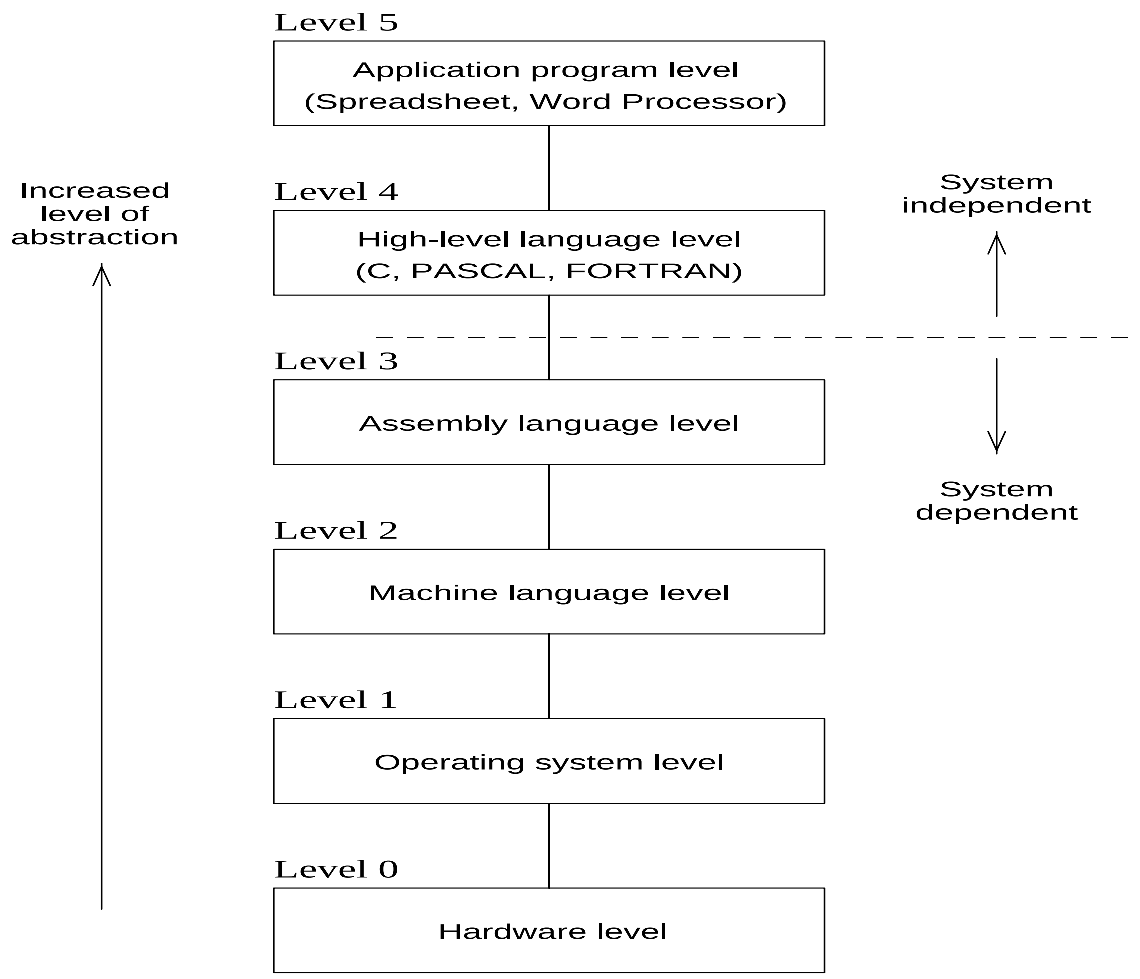
Outline

- A user's view of computer systems
- What is assembly language?
  - ∗ Relationship to machine language
- Advantages of high-level languages
  - ∗ Faster program development
  - ∗ Easier maintenance
  - ∗ Portability
- Why program in assembly language?
  - ∗ Time-efficiency
  - ∗ Space-efficiency
  - ∗ Accessibility to hardware
- Typical applications
- Why learn assembly language?
- Performance: C versus assembly language
  - ∗ Bubble sort example

A User’s View of Computer Systems

- Depends on the degree of abstraction provided by the underlying software
- We consider a hierarchy of five levels
  - ∗ Moving to the top of hierarchy shields the user from the lower-level details
  - ∗ The top two levels are *system independent*
  - ∗ The other lower four levels are *system dependent*
    - » Assembly and machine languages are specific to a particular processor
    - » One-to-one correspondence between assembly language and machine language

A User's View of Computer Systems (cont'd)



What is Assembly Language?

- Low-level language
  - » Each instruction performs a much lower-level task compared to a high-level language instruction
- One-to-one correspondence between assembly language and machine language instructions
  - » For most assembly language instructions, there is a machine language equivalent
  - » ***Assembler*** translates assembly language instructions to machine language instructions
- Directly influenced by the instruction set and architecture of the processor (CPU)

What is Assembly Language? (cont'd)

- Some example assembly language instructions:

```
inc    result
mov    class_size,45
and    mask1,128
add    marks,10
```

- Some points to note:
  - » Assembly language instructions are cryptic
  - » Mnemonics are used for operations
    - – **`inc`** for increment, **`mov`** for move (i.e., copy)
  - » Assembly language instructions are low level
    - – Cannot write instructions such as

```
mov    marks, value
```

What is Assembly Language? (cont'd)

- Some simple high-level language instructions can be expressed by a single assembly instruction

Assembly Language	**C**
`inc    result`	`result++;`
`mov    size,45`	`size = 45;`
`and    mask1,128`	`mask1 &= 128;`
`add    marks,10`	`marks += 10;`

What is Assembly Language? (cont'd)

- Most high-level language instructions need more than one assembly instruction

C	Assembly Language

`size = value;`	`mov AX,value`
	`mov size,AX`

`sum += x + y + z;`	`mov AX,sum`
	`add AX,x`
	`add AX,y`
	`add AX,z`
	`mov sum,AX`

What is Assembly Language? (cont'd)

- Readability of assembly language instructions is much better than the machine language instructions
  - » Machine language instructions are a sequence of 1s and 0s

Assembly Language		**Machine Language** (in Hex)
`inc`	`result`	`FF060A00`
`mov`	`class_size,45`	`C7060C002D00`
`and`	`mask,128`	`80260E0080`
`add`	`marks,10`	`83060F000A`

Advantages of High-Level Languages

- Program development is faster
  - » High-level instructions
    - – Fewer instructions to code
- Programs maintenance is easier
  - » For the same reasons as above
- Programs are portable
  - » Contain few machine-dependent details
    - – Can be used with little or no modifications on different types of machines
  - » Compiler translates to the target machine language
  - » Assembly language programs are not portable

Why Program in Assembly Language?

- Two main reasons:
  - ∗ Efficiency
    - » Space-efficiency
    - » Time-efficiency
  - ∗ Accessibility to system hardware
- Space-efficiency
  - ∗ Assembly code tends to be compact
- Time-efficiency
  - ∗ Assembly language programs tend to run faster
    - » Only a well-written assembly language program runs faster
      - – Easy to write an assembly program that runs slower than its high-level language equivalent

Typical Applications

- Application that need one of the three advantages of the assembly language
- Time-efficiency
  - ∗ Time-convenience
    - » Good to have but not required for functional correctness
      - – Graphics
  - ∗ Time-critical
    - » Necessary to satisfy functionality
    - » Real-time applications
      - – Aircraft navigational systems
      - – Process control systems
      - – Robot control software
      - – Missile control software

Typical Applications (cont'd)

- Accessibility to system hardware
  - ∗ System software typically requires direct control of the system hardware devices
    - » Assemblers, linkers, compilers
    - » Network interfaces, device drivers
    - » Video games
- Space-efficiency
  - ∗ Not a big plus point for most applications
  - ∗ Code compactness is important in some cases
    - – Portable and hand-held device software
    - – Spacecraft control software

Why Learn Assembly language?

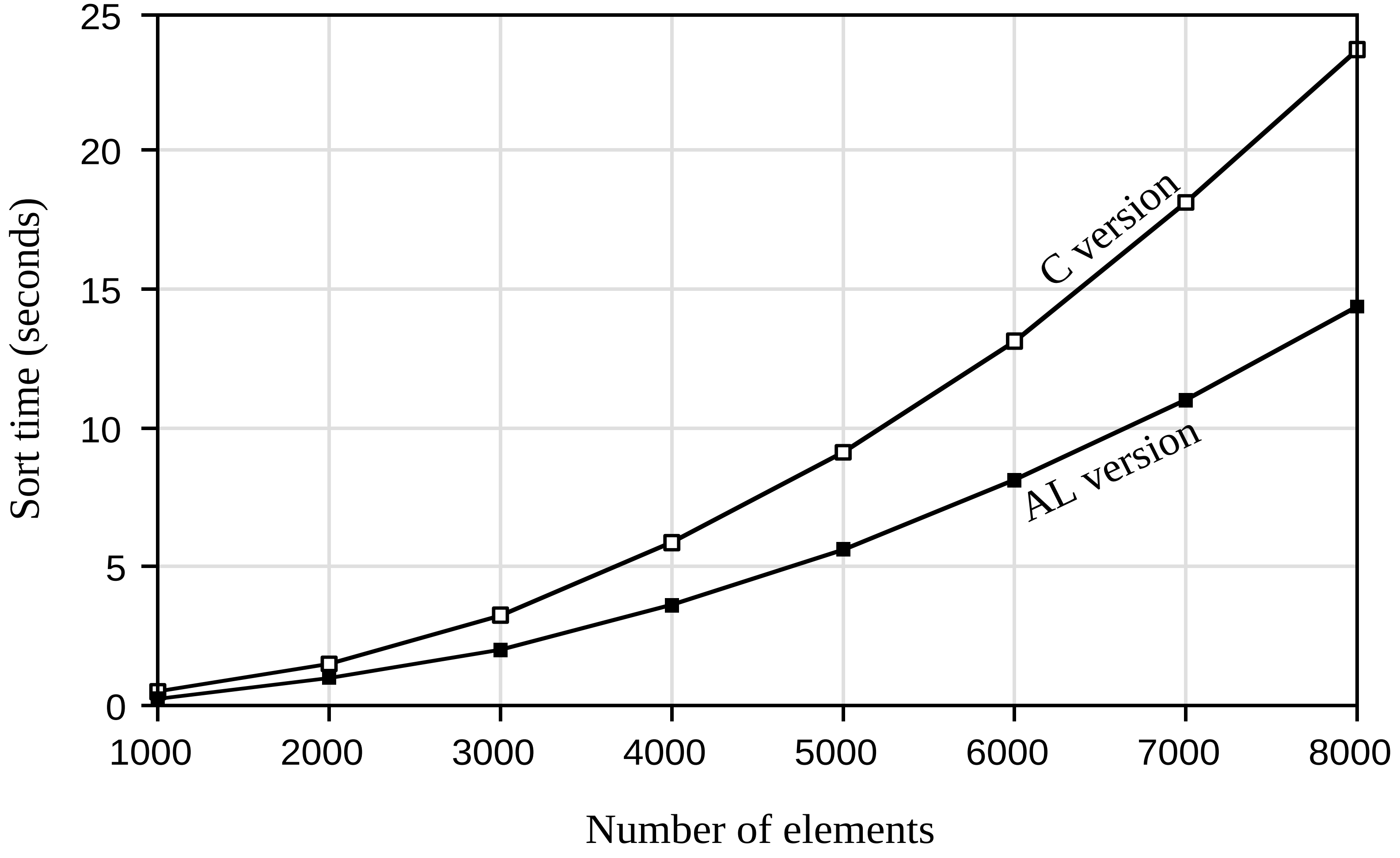
- Some applications require programming in assembly language
    - » Typically only a small part of an application is coded in assembly language (rest written in a high-level language)
        - – Such programs are called *mixed mode* programs
- Assembly language can be used as a tool to learn computer organization
    - » You will know more about the organization and internal workings of a computer system
- Personal satisfaction of learning something something complicated and useful

Performance: C versus Assembly Language

- We use bubble sort as an example
- Executable file size (space-efficiency)
  - C version: 50,256 bytes
  - Assembly version: 50,208 bytes
  - Negligible difference (only 48 bytes)

- Bubble sort procedure source code length
  - C version: 1,340 bytes
  - Assembly version: 1,851 bytes
  - Shows the low-level nature of the assembly code

Performance: C versus Assembly Language (cont'd)


Sort time (seconds)
Number of elements
C version
AL version