02A System Software
CSC 230

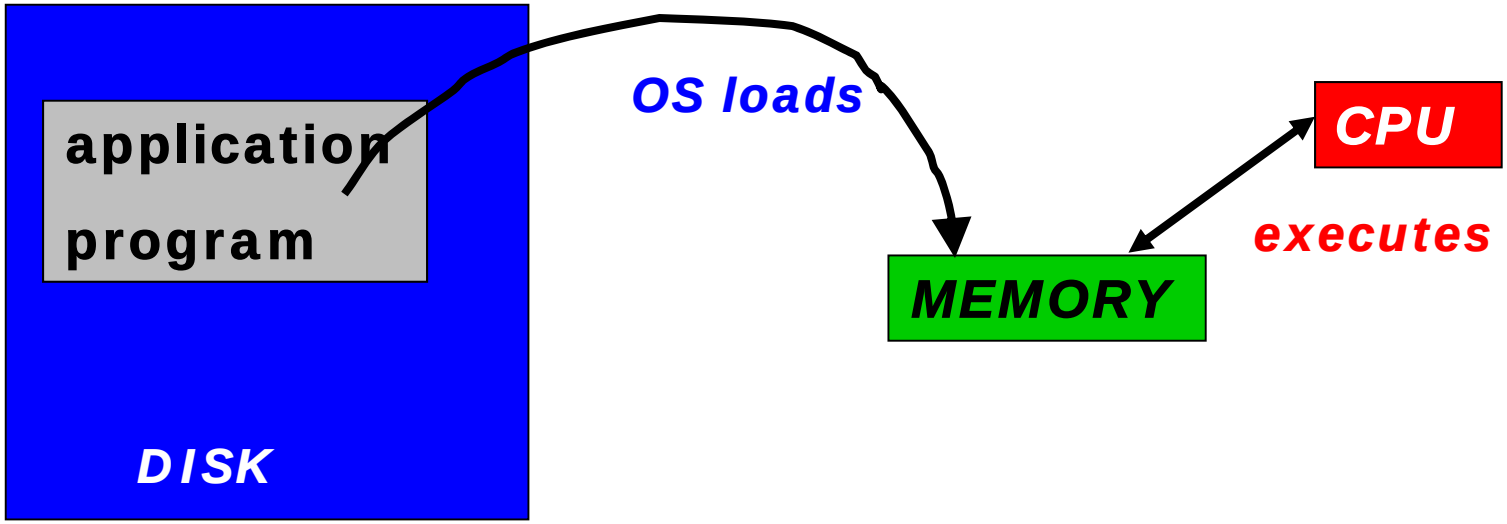
Department of Computer Science
University of Victoria

Stl: 2.1; 2.2; (2.3 and 2.4 read); 2.5; part of 3.4;

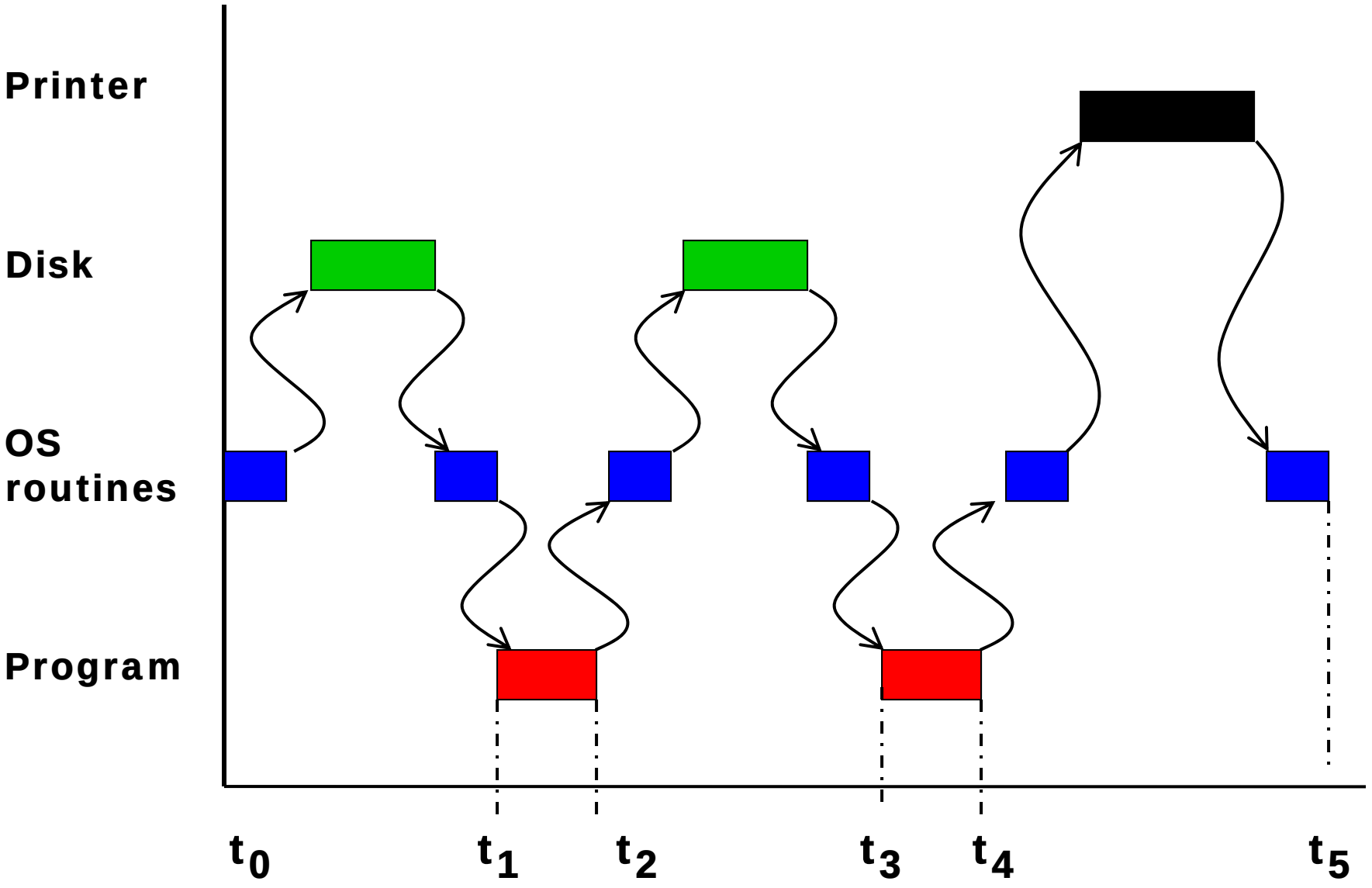
(Optional M&H: 7.1; 6.1)

System Software

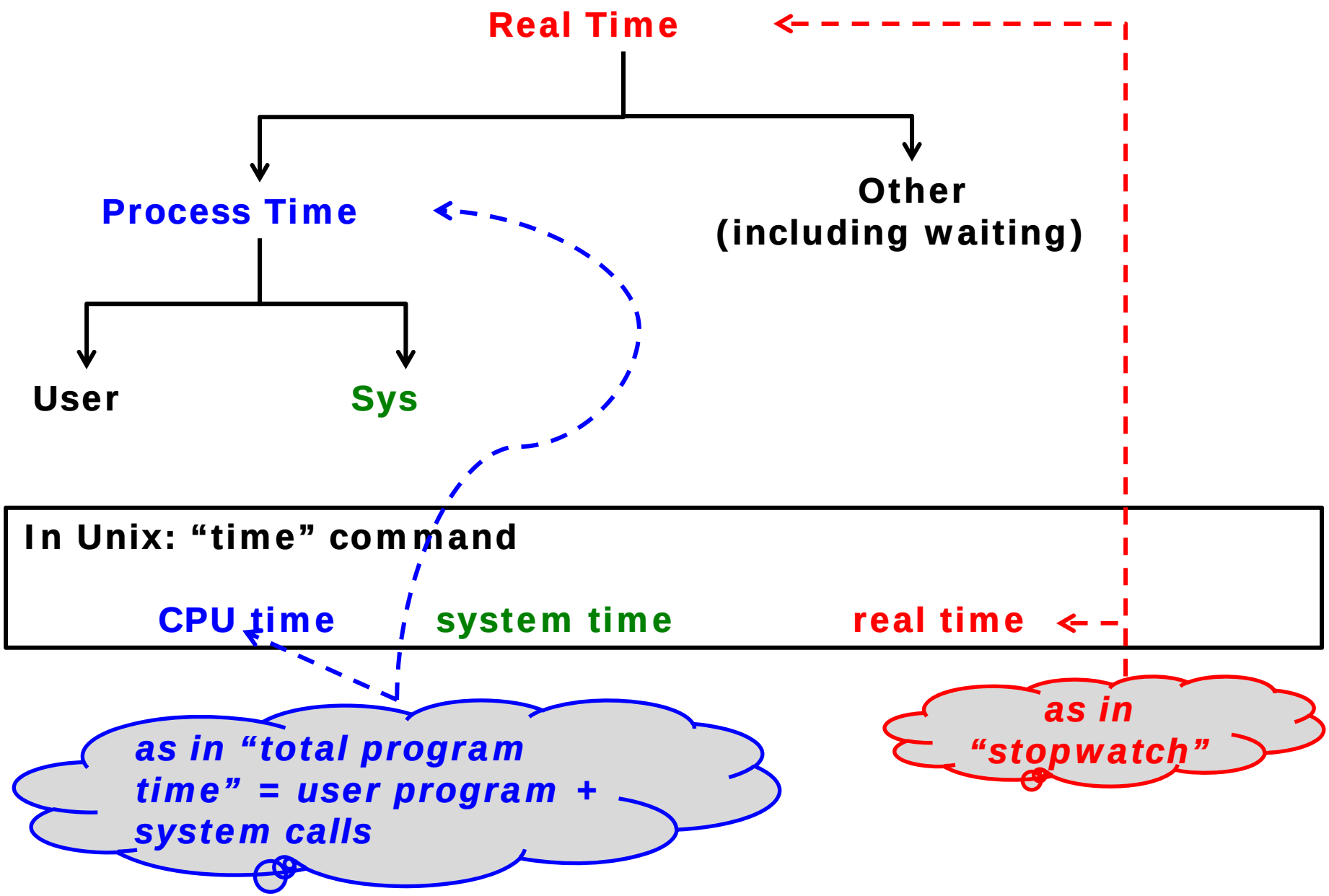
- collection of programs
- interpreting user commands
- managing storage and retrieval of files from storage
- controlling I/O units
- linking and loading and executing user programs
- OS = key system software component → to control sharing and interaction among computer units while executing applications



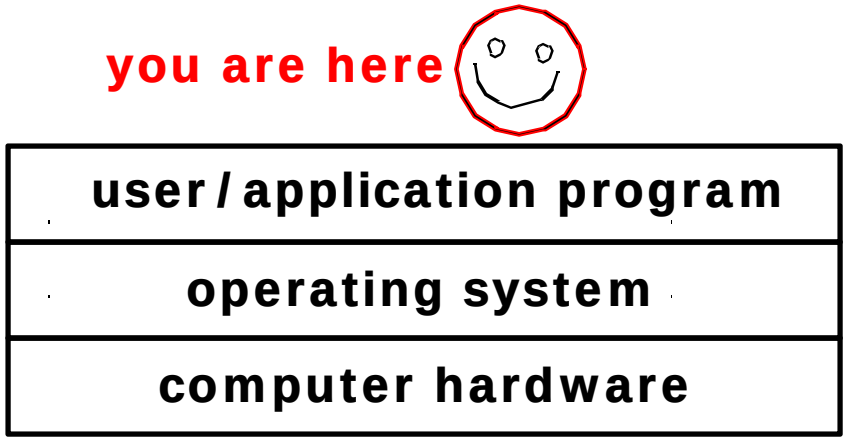
System Software



Real versus Process (user/sys) Time

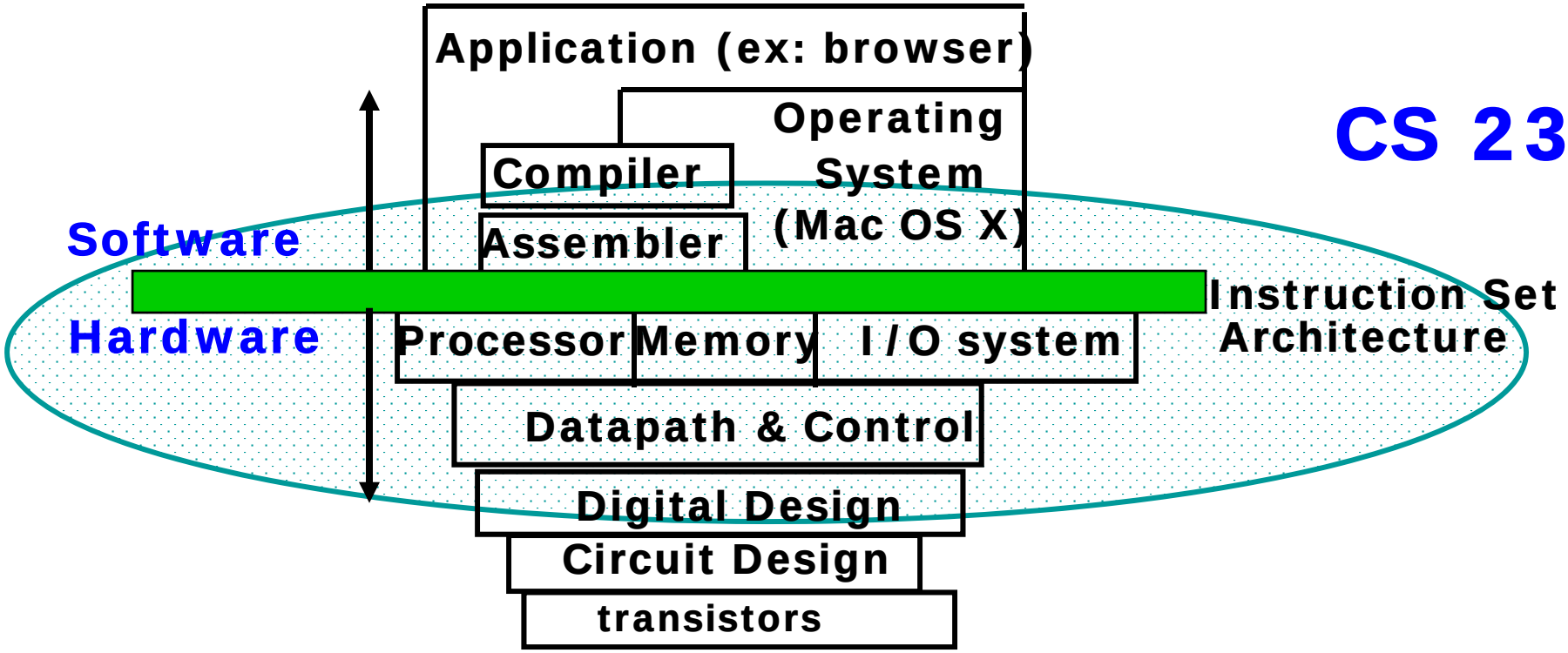


Typical structure



- middle level hides the underlying hardware
- presents multiple users and multiple tasks with the appearance that each is in sole possession of the machine
- each has a *virtual* or *abstract* machine on which to run.

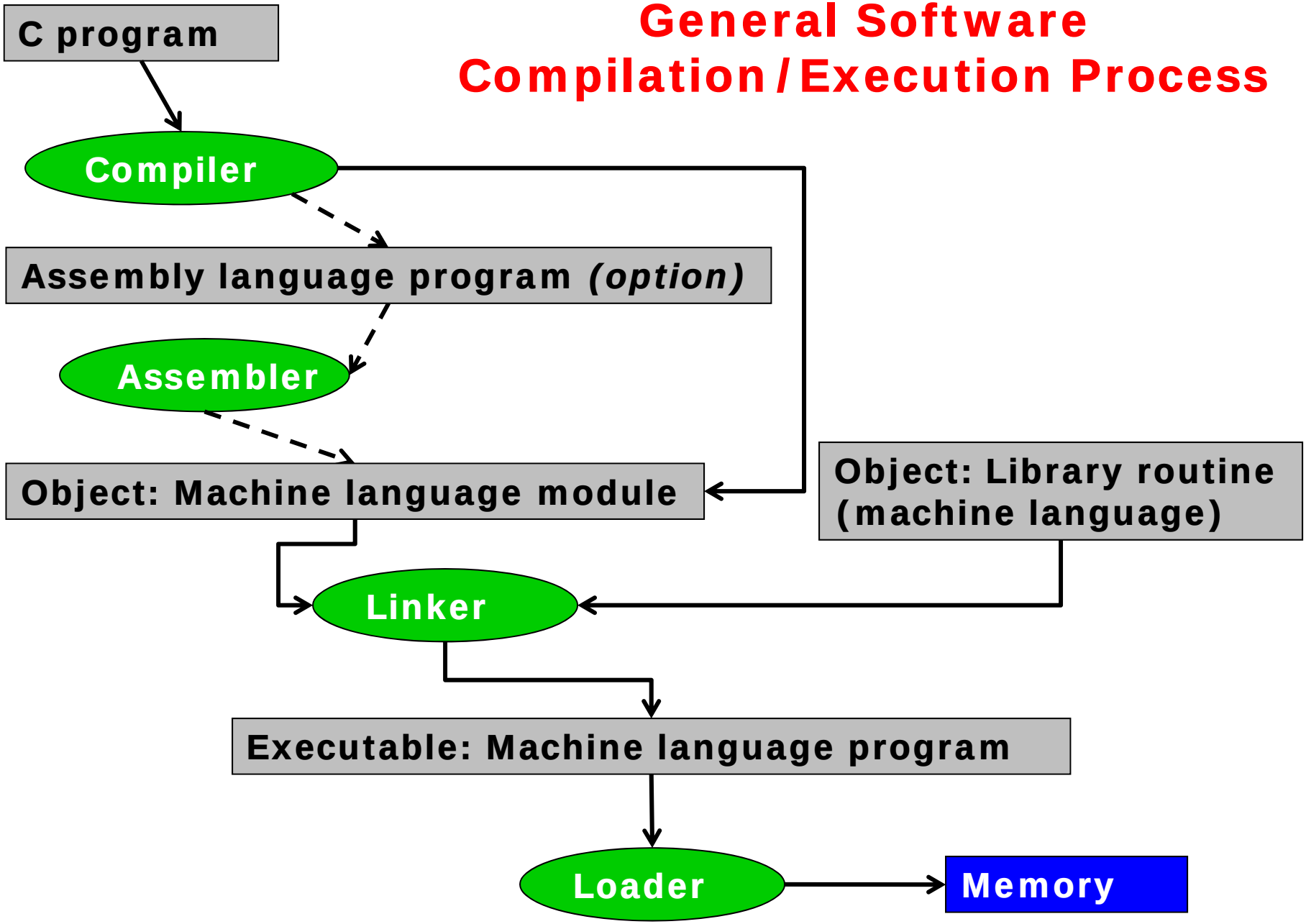
What are "Machine Structures"?



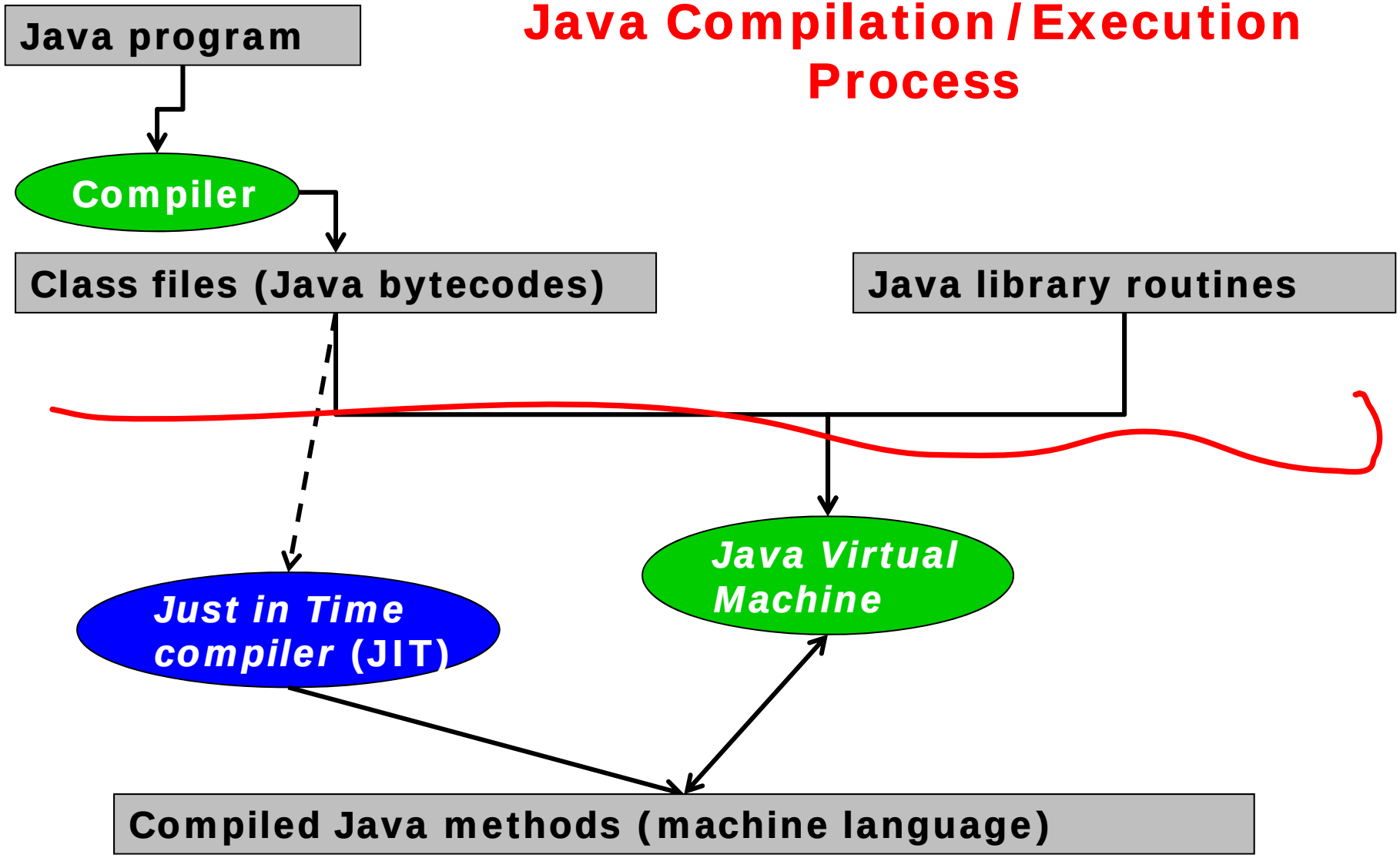
CS 230

* Coordination of many *levels (layers) of abstraction*

General Software Compilation / Execution Process



Java Compilation / Execution Process



Which languages are compiled-based?
Which run on a Virtual Machine?
Which are Object Oriented?

C

C++

C#

Fortran

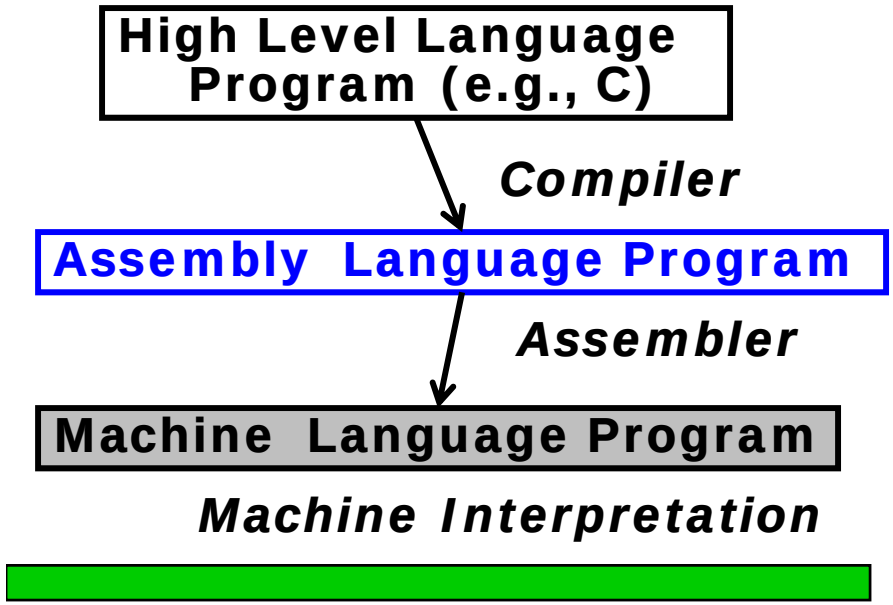
Java

Python

Perl

Others?

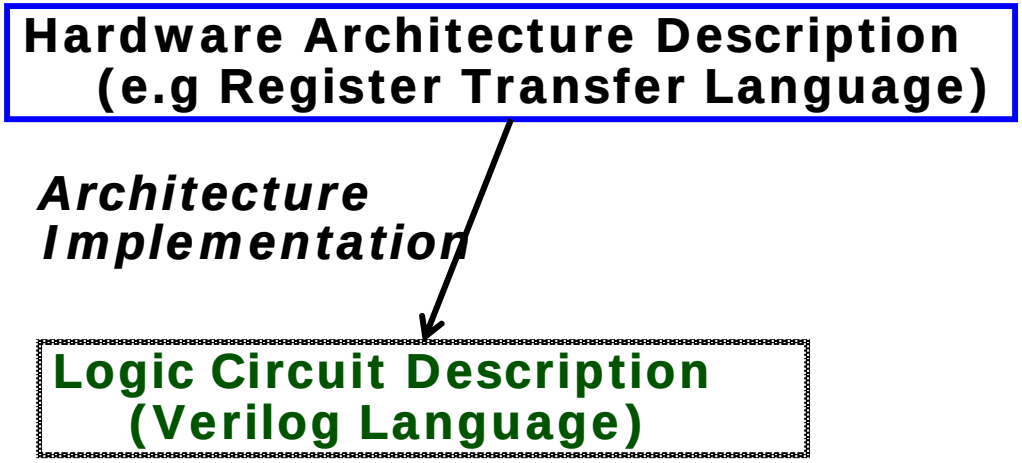
Levels of Representation




```
temp = v[k];
v[k] = v[k+1];
v[k+1] = temp;
```

```
ldr  R0, [R1]
ldr  R2, [R1,#4]
str  R0, [R1,#4]
str  R2, [R1]
```

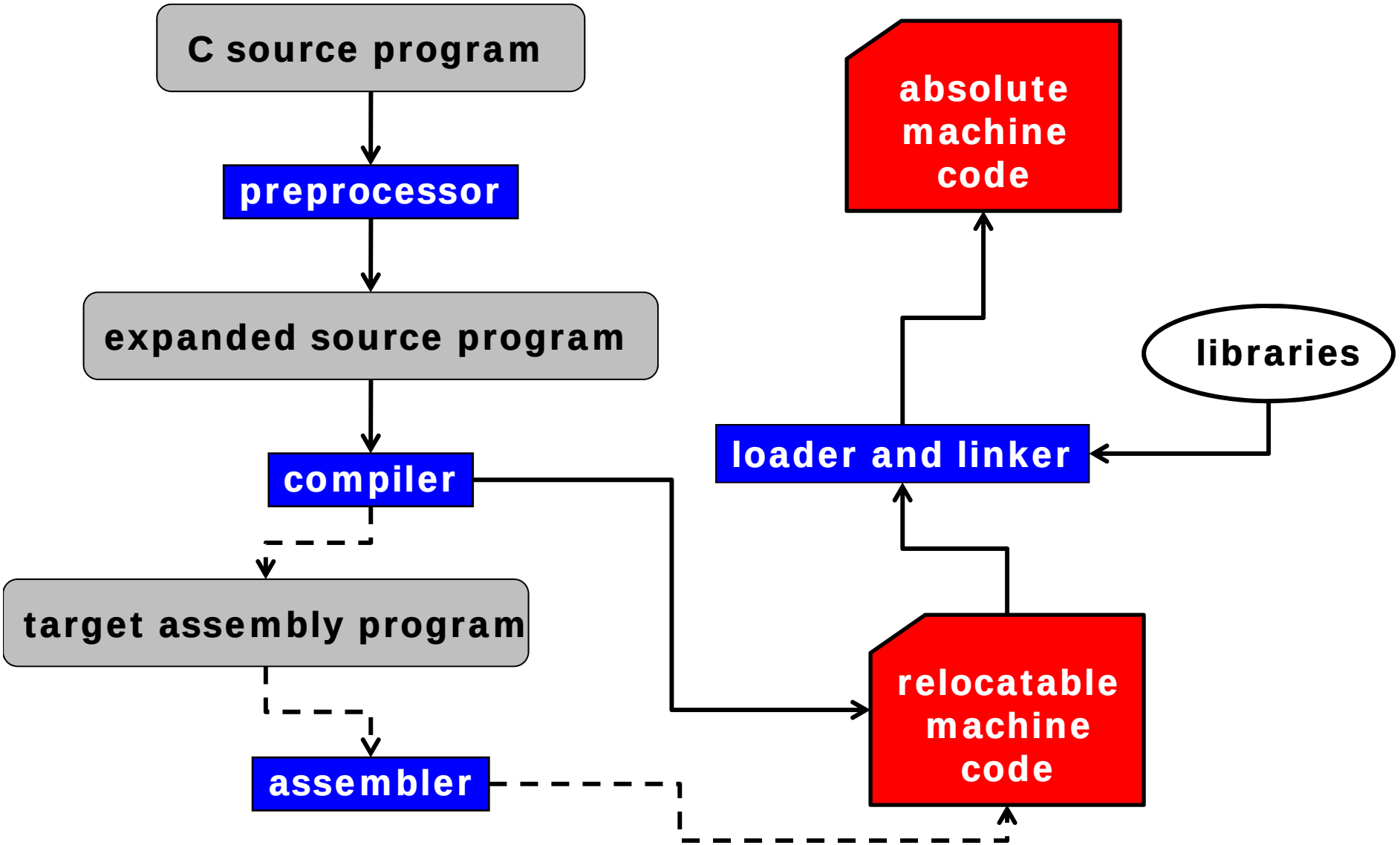
```
1110 0101 1001 0001 0000 0000 0000 0000
1110 0101 1001 0001 0010 0000 0000 0100
1110 0101 1000 0001 0000 0000 0000 0100
1110 0101 1000 0001 0010 0000 0000 0000
```



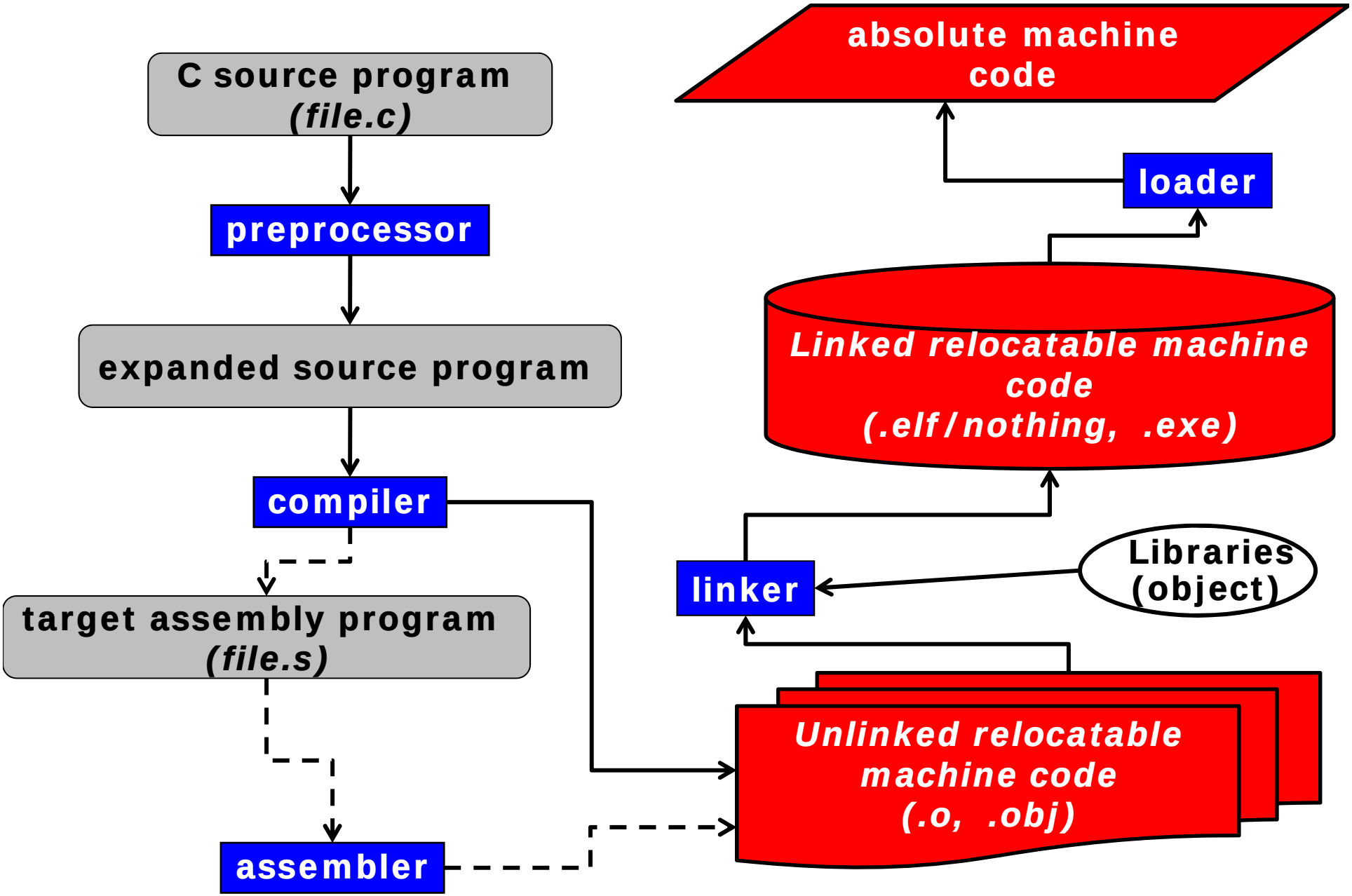

```
wire [31:0] dataBus;
regFile registers (databus);
ALU ALUBlock (inA, inB,
   databus);
```

```
wire w0;
XOR (w0, a, b);
AND (s, w0, a);
```

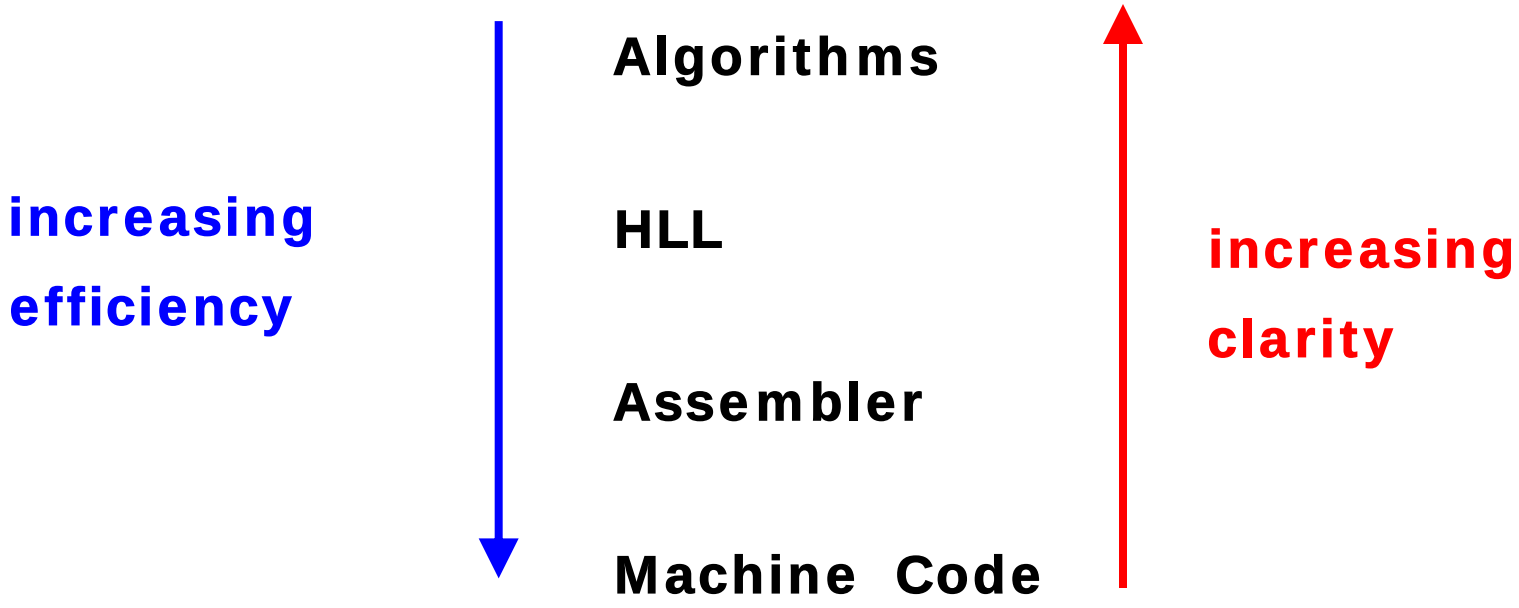
Example Language Processing System



Example Language Processing System



Computer Language Hierarchy



Machine code is entirely numeric.
It is the language level directly executed by a computer.
Hence, machine code is unique to the computer design, more specifically it is unique to the design of the central processing unit.

Assembly Language

More readable symbolic version of numeric machine code

One to one correspondence of assembly language instructions to machine language instructions

ldr R0, [R1]	⟷	**1110 0101 1001 0001 0000 0000 0000 0000** **E591 0000**
ldr R2, [R1,#4]	⟷	**1110 0101 1001 0001 0010 0000 0000 0100** **E591 2004**
str R0, [R1,#4]	⟷	**1110 0101 1000 0001 0000 0000 0000 0100** **E581 0004**
str R2, [R1]	⟷	**1110 0101 1000 0001 0010 0000 0000 0000** **E581 2000**

- ***Why bother if we have high level languages?***
  - **for access to system instructions which are not used by the compiler**
  - **for cases where speed/timing is very important (e.g. real time environments)**
  - **for cases where there is nothing else available**
- ***Reason to study assembler language***
  - **understanding architectural and language issues**
- **Good approach in general:**

often >90% of time spent executing <10% of the code

(a) write in HLL → (b) do an execution profile → (c) recode critical parts in assembly language

Compilers generate general code for all sorts of applications - whereas particular cases may need optimization at a fine grain level

What are? Find out!

Hardware ***Software*** ***Firmware***

Programming is preparing the list of instructions to be executed

A program is loaded into memory

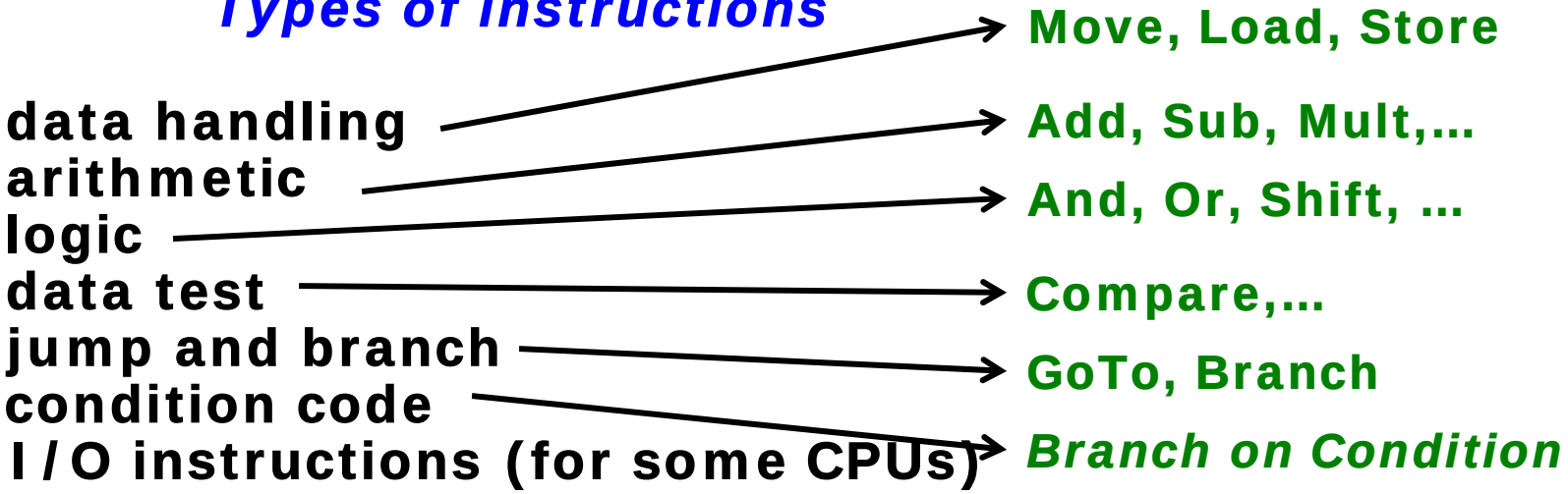
Data is loaded into memory and is manipulated by the program.

Data and program memory spaces may be together or separate

Basics of Machine Language

- numerical instructions specific to the particular CPU
- von Neumann's concept? ➔ stored program execution

Types of instructions



Layout of each instruction content :

1) operation code (opcode)
2) zero or more operands
   (data or locations to be operated upon)

John von Neumann in 1945

He stated what a general-purpose electronic computer ought to be like

A computer has a "von Neumann architecture" if :

- ✓ **It consists of ALU, control unit, memory, and I/O devices.**
- ✓ **The memory just stores numbers (integers of limited size).**
- ✓ **The program is encoded numerically and stored in the memory along with the data.**
- ✓ **One instruction is executed at a time. It may cause the transfer of a bounded (and small) amount of data to and from the memory and I/O devices.**