ASSEMBLY LANGUAGE

BSCS/BSSE

Computing Machines

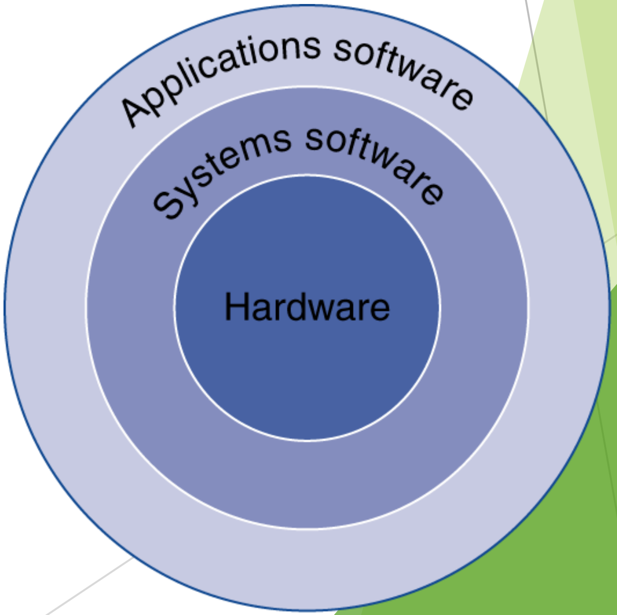
- *General purpose*: servers, desktops, laptops, PDAs, etc.
- *Special purpose*: cash registers, ATMs, games, Mobile Phones, etc.
- *Embedded*: cars, door locks, printers, digital players, industrial machinery, medical equipment, etc.

Distinguishing Characteristics

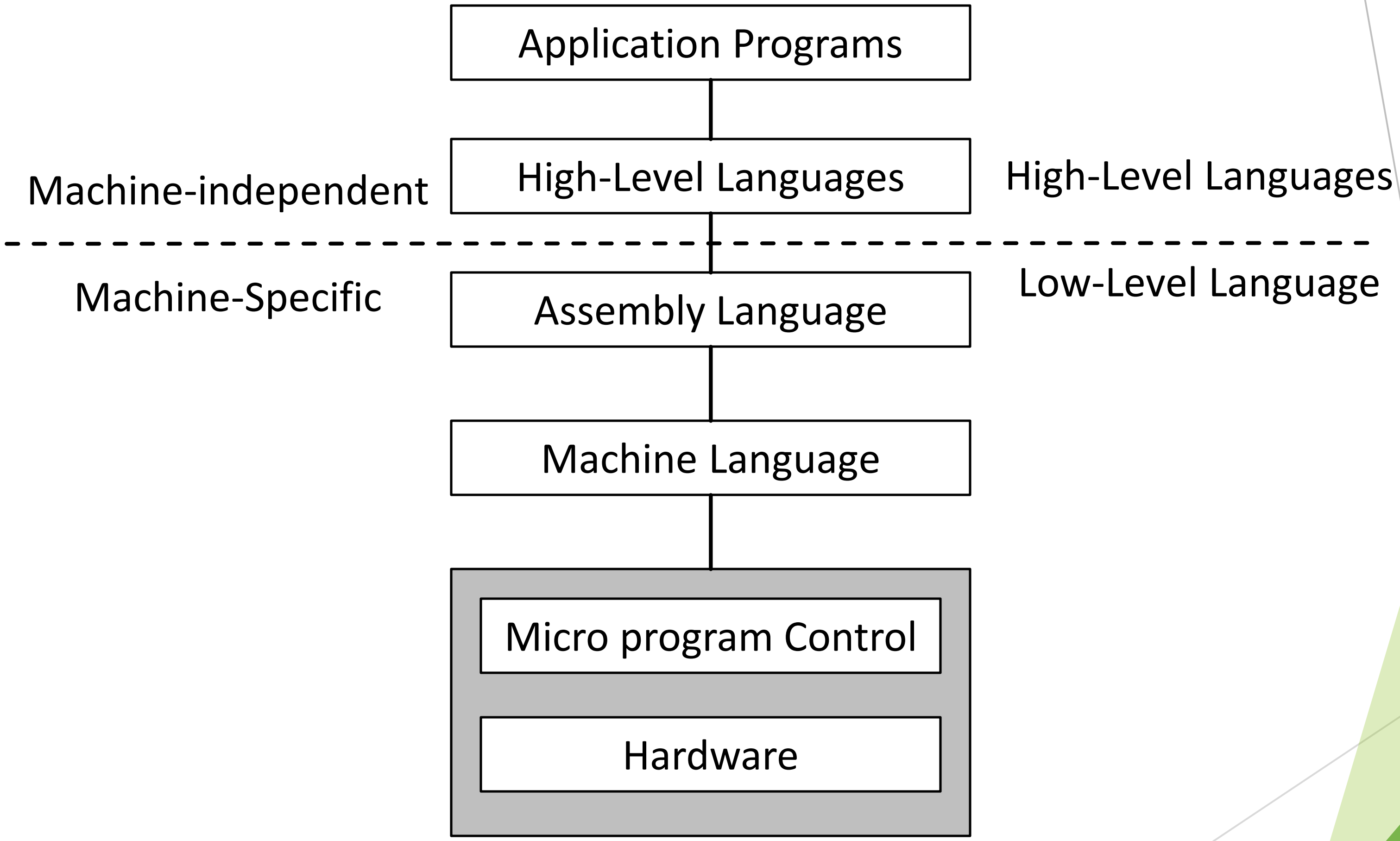
- Speed
- Cost
- Ease of use, software support & interface
- Scalability

Inside the Computer

- **Application software**
  - Written in high-level language
- **System software**
  - Compiler: translates HLL code to machine code
  - Operating System: service code
    - Handling input/output
    - Managing memory and storage
    - Scheduling tasks & sharing resources
- Hardware
  - Processor, memory, I/O controllers

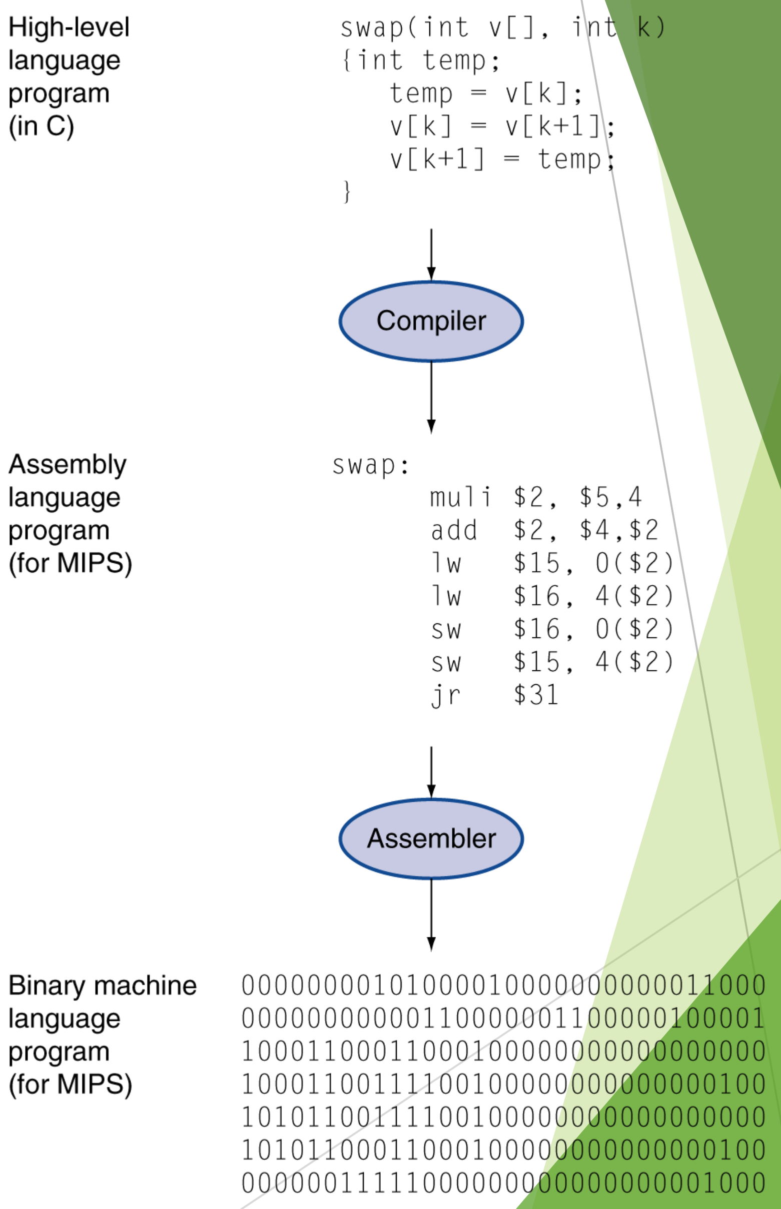


A Programmer's View of a Computer



Levels of Program Code

- **High-level language**
  - Level of abstraction closer to problem domain
  - Provides productivity and portability
- **Assembly language**
  - Textual representation of instructions
- **Hardware representation**
  - Binary digits (bits)
  - Encoded instructions and data



High-level language program (in C)

```
swap(int v[], int k)
{int temp;
   temp = v[k];
   v[k] = v[k+1];
   v[k+1] = temp;
}
```

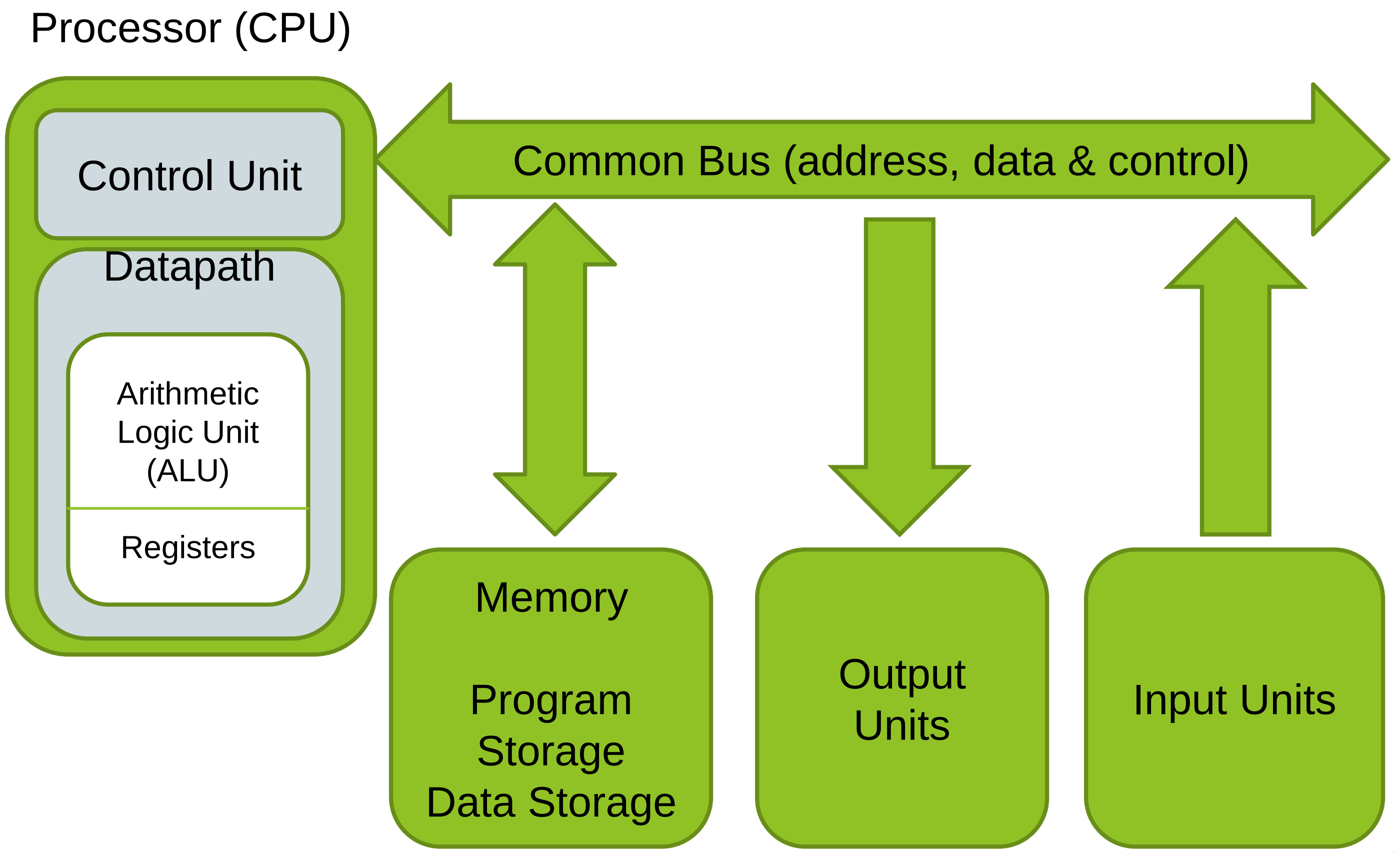
Assembly language program (for MIPS)

```
swap:
      muli $2, $5,4
      add  $2, $4,$2
      lw   $15, 0($2)
      lw   $16, 4($2)
      sw   $16, 0($2)
      sw   $15, 4($2)
      jr   $31
```

Binary machine language program (for MIPS)

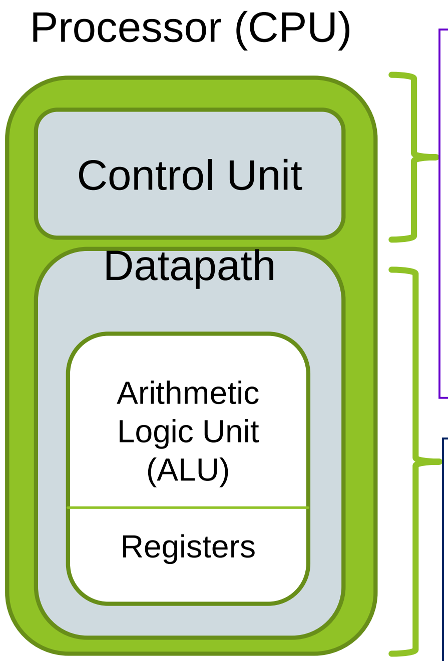
```
00000000101000010000000000011000
00000000000110000001100000100001
10001100011000100000000000000000
10001100111100100000000000000100
10101100111100100000000000000000
10101100011000100000000000000100
00000011111000000000000000001000
```

Detailed Anatomy of a Computer



Anatomy of a Computer: *CPU*

The brain of a Computer System



•Decodes and monitors the execution of instructions.
•Controls flow of information in CPU, memory, I/O devices:

•System clock (Intel® Core™ I7-720QM Processor (1.6GHz, turbo up to 2.8GHz, 6MB L3 Cache))
•Maintains a register called program counter(PC)

•**ALU**: performs all arithmetic computations & logic evaluations.
•**Registers**: storage location in CPU, used to hold data or a memory address during the execution of an instruction..

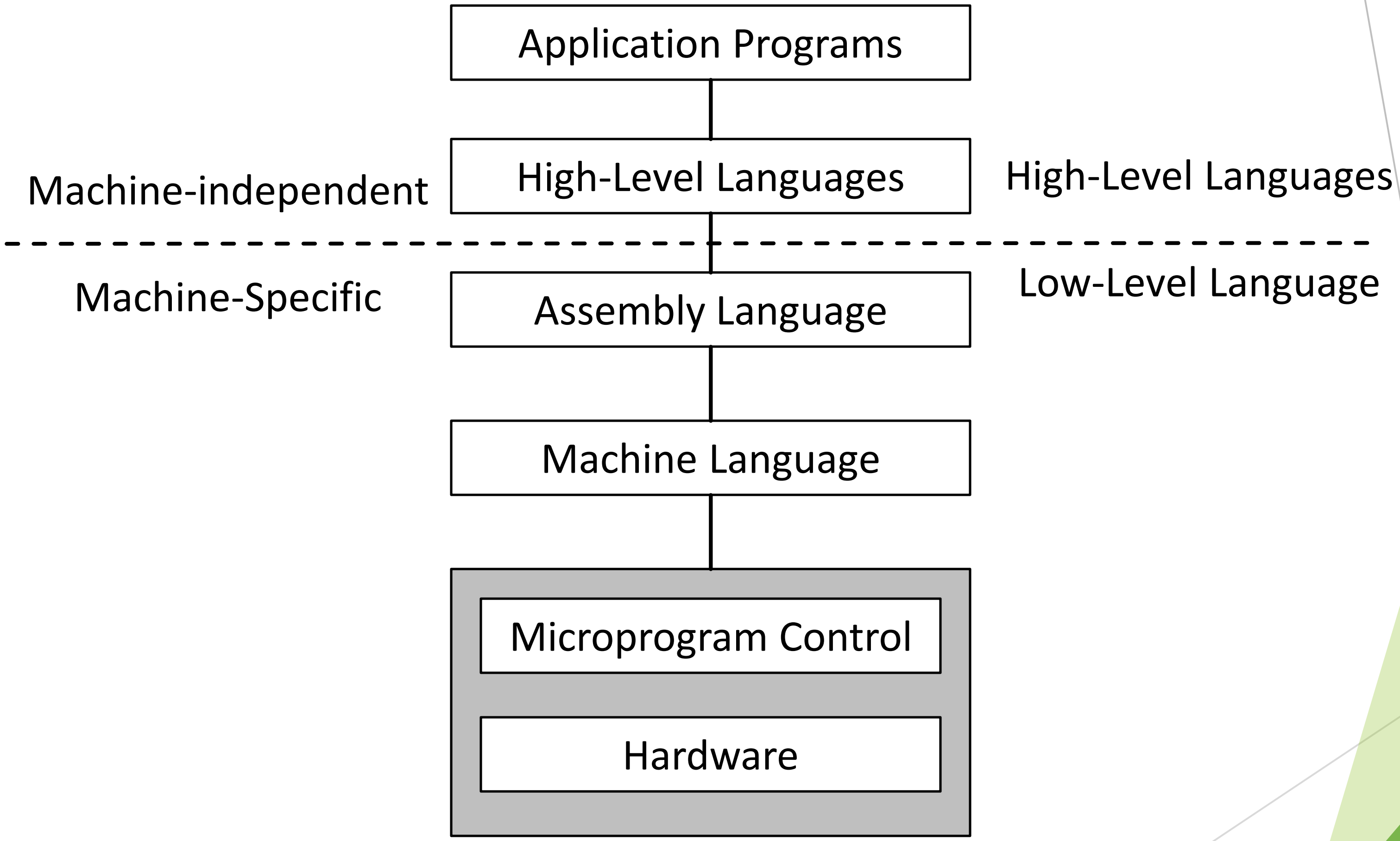
Anatomy of a Computer: *Common Bus*

Common Bus (address, data & control)

A group of conducting wires that allow signals to travel from one point to another:

- Address bus: the location of data in memory or I/O devices
- Data bus: carry data in & out from CPU
- Control bus: control the operation of the CPU

A Hierarchy of Languages



Application Programs

High-Level Languages

Machine-independent

High-Level Languages

Machine-Specific

Assembly Language

Low-Level Language

Machine Language

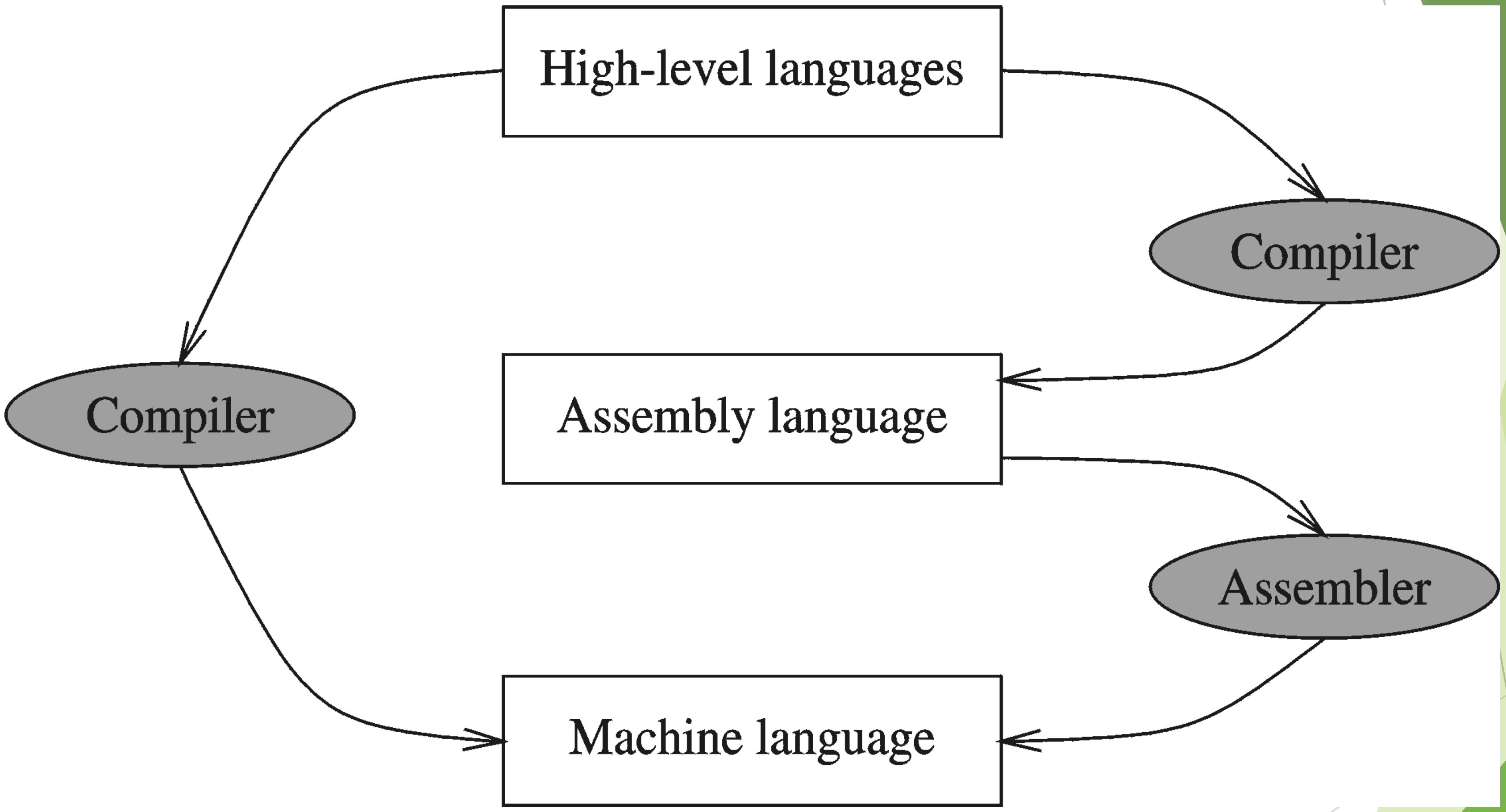
Microprogram Control

Hardware

Assembly and Machine Language

- Machine language
  - Native to a processor: executed directly by hardware
  - Instructions consist of binary code: 1s and 0s
- Assembly language
  - A programming language that uses symbolic names to represent operations, registers and memory locations.
  - Slightly higher-level language
  - Readability of instructions is better than machine language
  - One-to-one correspondence with machine language instructions
- Assemblers translate assembly to machine code
- Compilers translate high-level programs to machine code
  - Either directly, or
  - Indirectly via an assembler

Compiler and Assembler



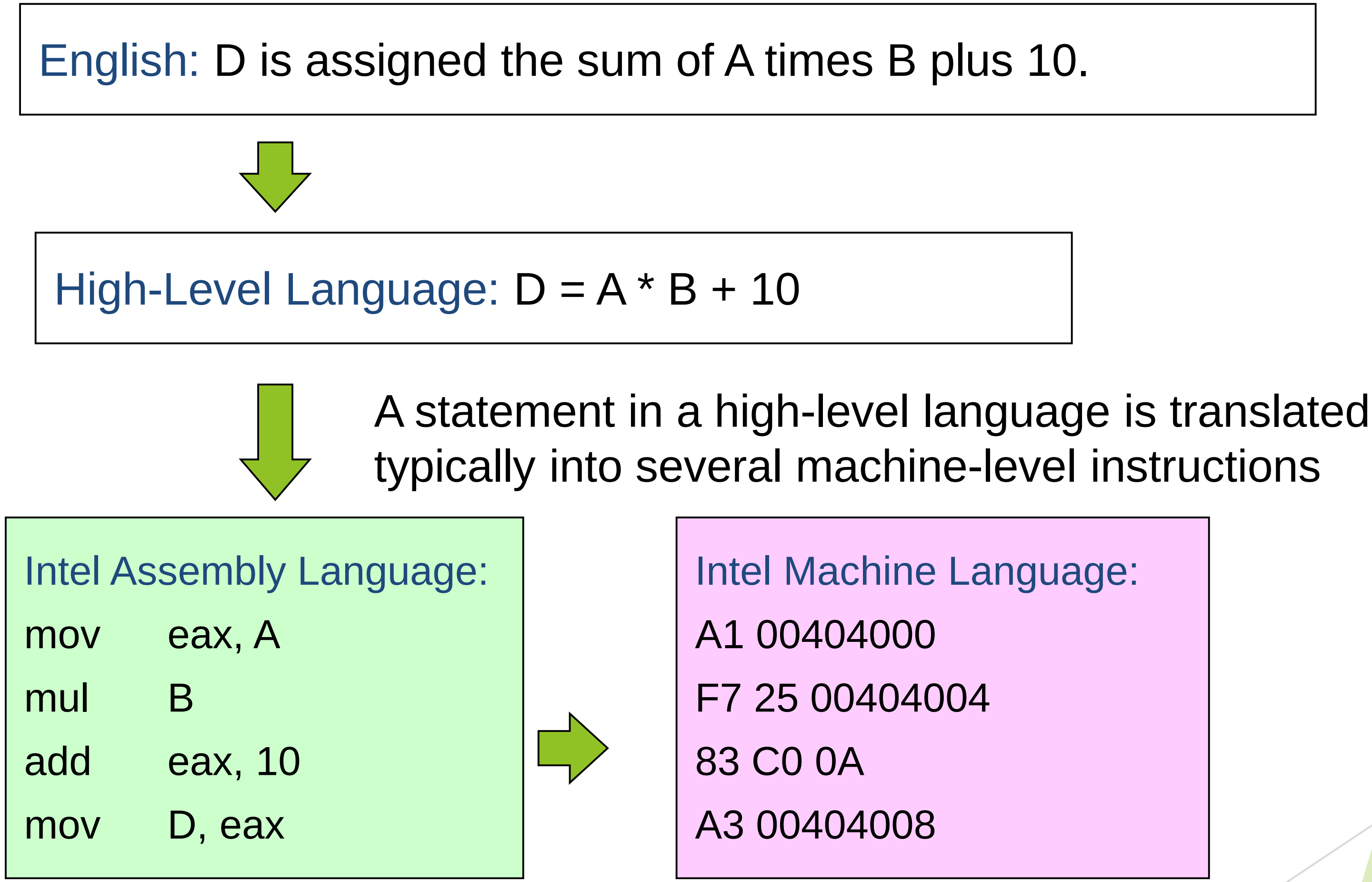
Instructions and Machine Language

- Each command of a program is called an ***instruction*** (it instructs the computer what to do).
- Computers only deal with binary data; hence the instructions must be in binary format (0s and 1s) .
- The set of all instructions (in binary form) makes up the computer's ***machine language***. This is also referred to as the ***instruction set***.

Instruction Fields

- Machine language instructions usually are made up of several fields. Each field specifies different information for the computer. The major two fields are:
- ***Opcode*** field which stands for operation code and it specifies the particular operation that is to be performed.
  - Each operation has its unique opcode.
- ***Operands*** fields which specify where to get the source and destination operands for the operation specified by the opcode.
  - The source/destination of operands can be a constant, the memory or one of the general-purpose registers.

Translating Languages



English: D is assigned the sum of A times B plus 10.

High-Level Language: D = A * B + 10

A statement in a high-level language is translated typically into several machine-level instructions

Intel Assembly Language:

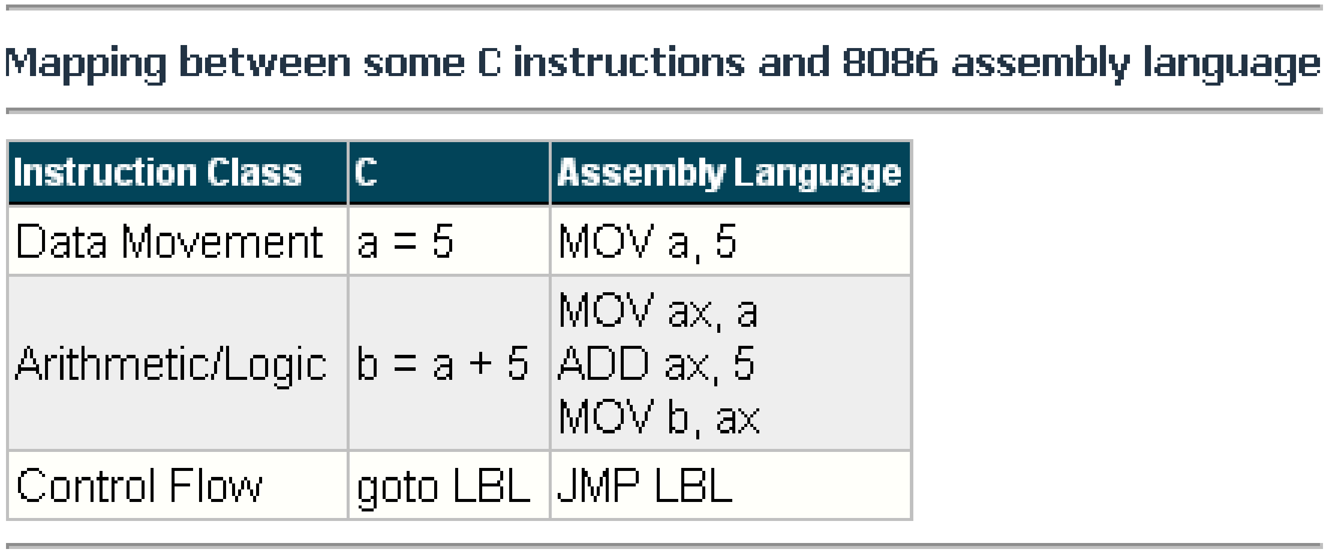
```
mov     eax, A
mul     B
add     eax, 10
mov     D, eax
```

Intel Machine Language:

```
A1 00404000
F7 25 00404004
83 C0 0A
A3 00404008
```

Mapping Between Assembly Language and HLL

- Translating HLL programs to machine language programs is not a one-to-one mapping
- A HLL instruction (usually called a statement) will be translated to one or more machine language instructions



Mapping between some C instructions and 8086 assembly language

Instruction Class	C	Assembly Language
Data Movement	a = 5	MOV a, 5
Arithmetic/Logic	b = a + 5	MOV ax, a ADD ax, 5 MOV b, ax
Control Flow	goto LBL	JMP LBL

Advantages of High-Level Languages

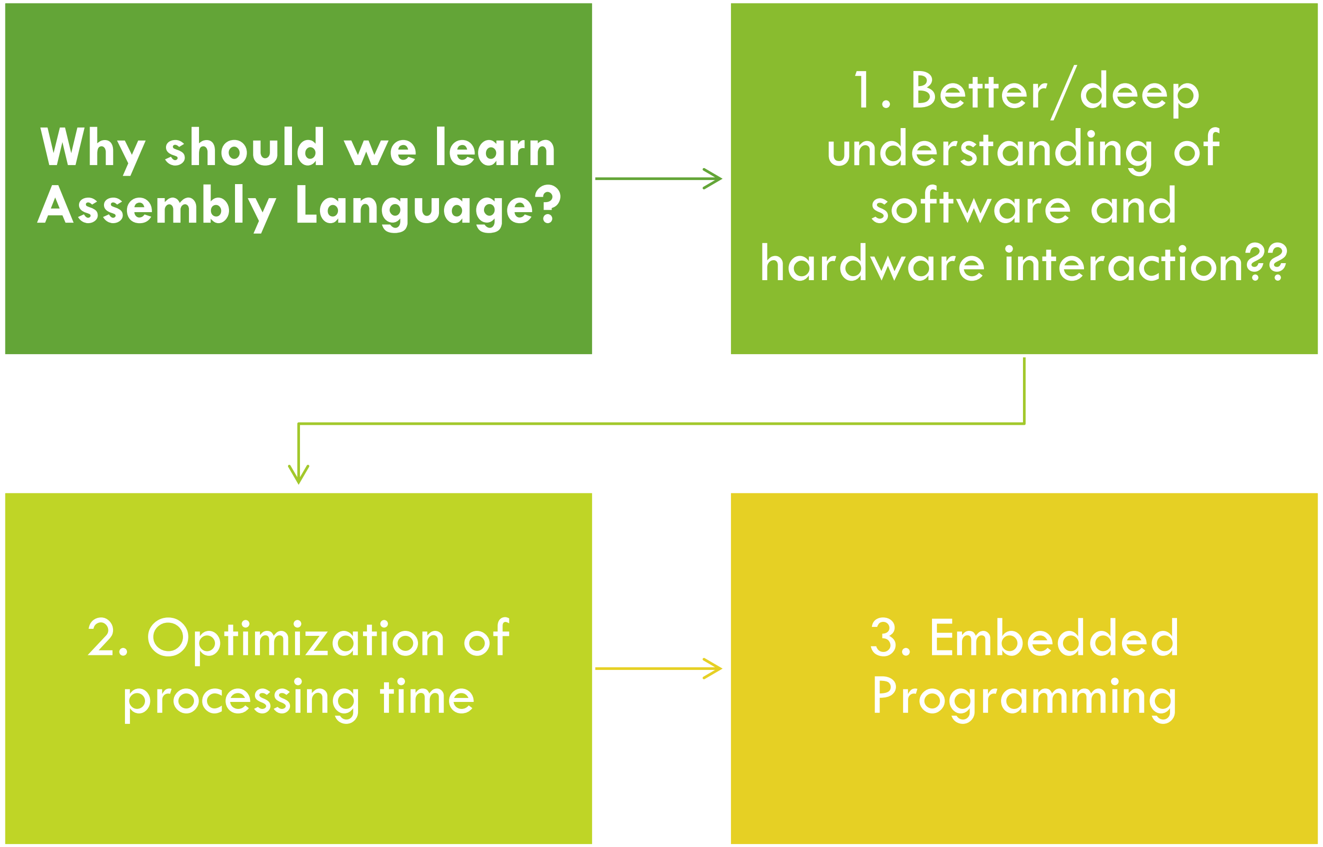
- Program development is faster
  - High-level statements: fewer instructions to code
- Program maintenance is easier
  - For the same above reasons
- Programs are portable
  - Contain few machine-dependent details
    - Can be used with little or no modifications on different machines
  - Compiler translates to the target machine language
  - However, Assembly language programs are not portable

What is Assembly Language?

Developed by David John Wheeler

(to interact with machine language easier to perform task).

- 1. Computer Programming Language
- 2. Low level Programming Language
- It uses Mnemonics/keywords
- Closer to Hardware
- Time waste in compilation can reduced

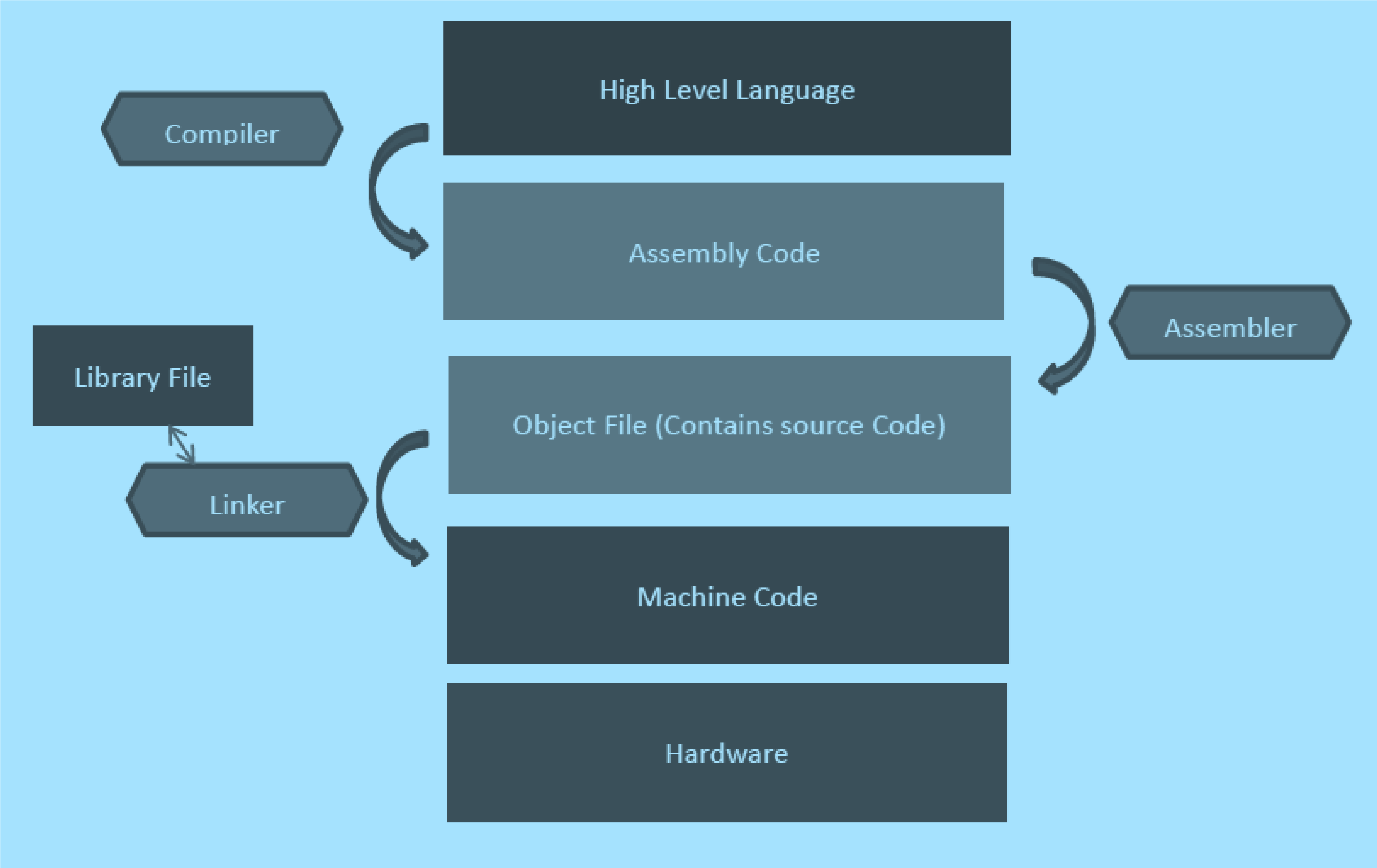


Why should we learn Assembly Language?

1. Better/deep understanding of software and hardware interaction??

2. Optimization of processing time

3. Embedded Programming



High Level Language

Compiler

Assembly Code

Assembler

Library File

Object File (Contains source Code)

Linker

Machine Code

Hardware

1. High level language program(C,C++, etc.,), Compiler(language translator) is used to convert high level code into Assembly.

2. Assembly code is then converted into object file by Assembler.

3. Then object file convert into machine code. Object file is linked by linker with library file. Every system has different system properties(how is it made, attributes, etc.). This library file present in system and object file link together change into machine code which run the hardware.

Code in C, filename.c (.c extension language format)

Compile in Assembly language, extension changes to filename.asm

Assembler convert it into object file, extension changes into filename.obj

Linker link it with library file extension changes into .exe, hardware run it.

Why Learn Assembly Language?

- Accessibility to system hardware
  - Assembly Language is useful for implementing system software
  - Also useful for small embedded system applications
- Space and Time efficiency
  - Understanding sources of program inefficiency
  - Tuning program performance
  - Writing compact code
- Writing assembly programs gives the computer designer the needed deep understanding of the instruction set and how to design one
- To be able to write compilers for HLLs, we need to be expert with the machine language. Assembly programming provides this experience

Assembly vs. High-Level Languages

- ***HLL programs*** are machine independent. They are easy to learn and easy to use.

- ***Assembly language programs*** are machine specific. It is the language that the processor directly understands.

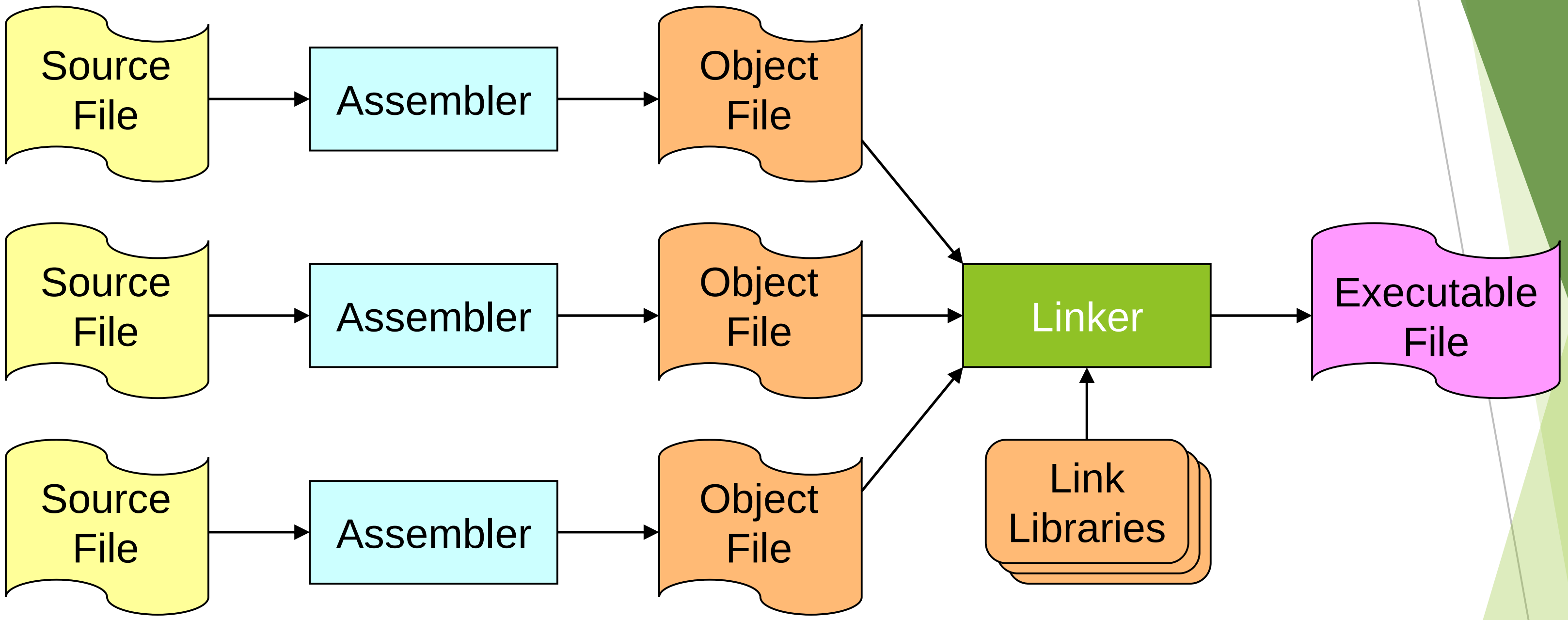
Tools for Assembly Language: *Assembler*

- Software tools are needed for editing, assembling, linking, and debugging assembly language programs
- An assembler is a program that converts source-code programs written in assembly language into object files in machine language
- Popular assemblers have emerged over the years for the Intel family of processors. These include …
  - TASM (Turbo Assembler from Borland)
  - NASM (Netwide Assembler for both Windows and Linux), and
  - GNU assembler distributed by the free software foundation
  - MASM (Macro Assembler from Microsoft)

Tools for Assembly Language: *Linker & Libraries*

- You need a linker program to produce executable files
- It combines your program's object file created by the assembler with other object files and link libraries, and produces a single executable program

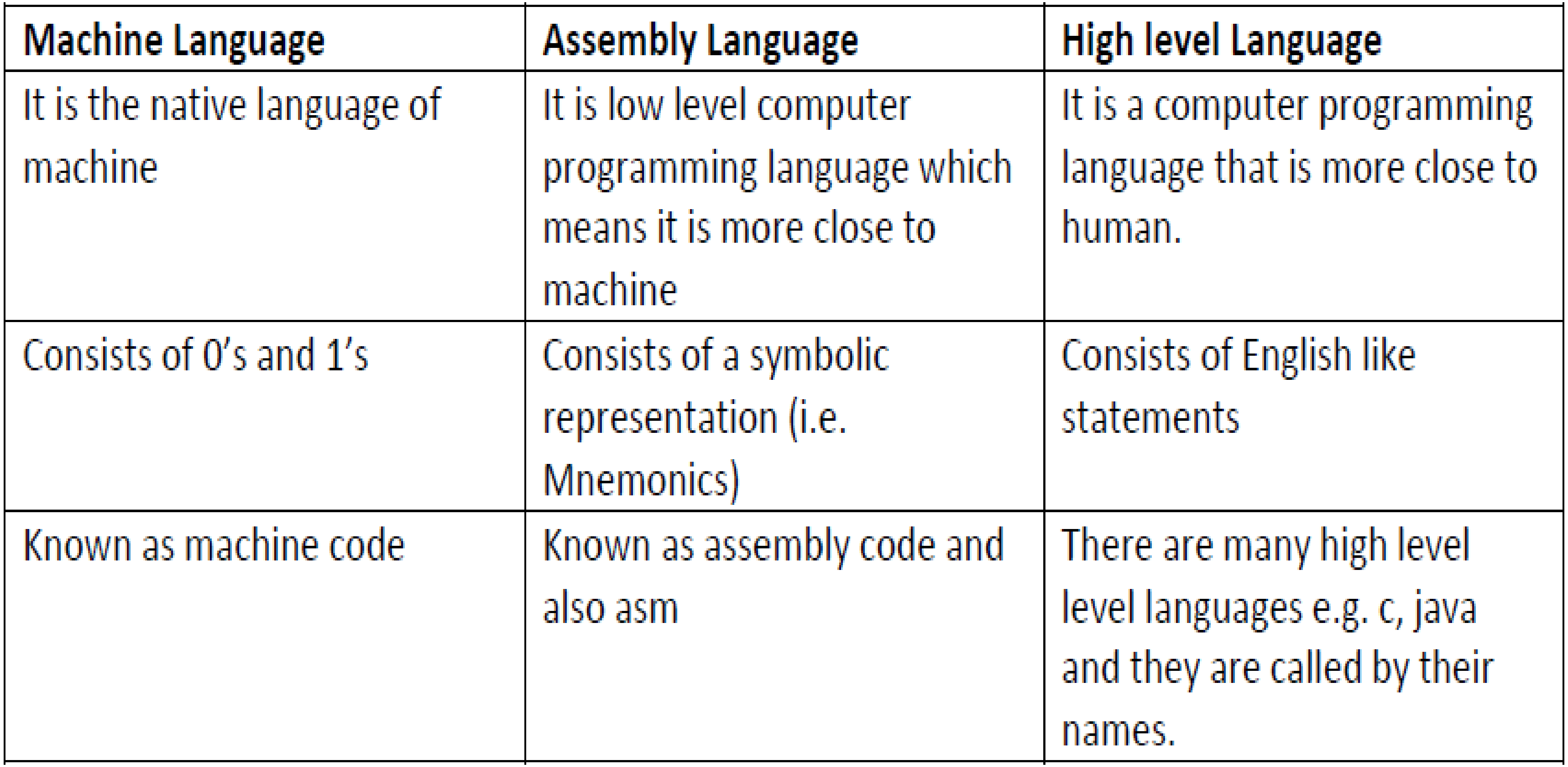
Assemble and Link Process



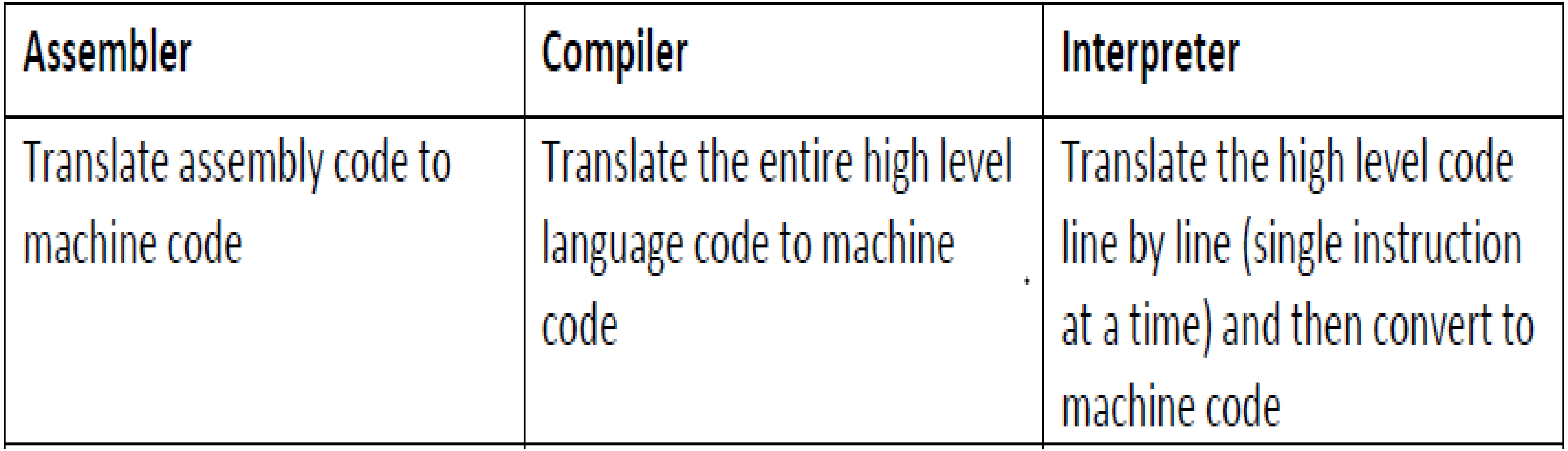
- A project may consist of multiple source files
- Assembler translates each source file separately into an object file
- Linker links all object files together with link libraries

REASON

Reason was to combine machine parts to perform specific task (like switches and other components which assemble to perform task), name Assembly



Machine Language	**Assembly Language**	**High level Language**
It is the native language of machine	It is low level computer programming language which means it is more close to machine	It is a computer programming language that is more close to human.
Consists of 0's and 1's	Consists of a symbolic representation (i.e. Mnemonics)	Consists of English like statements
Known as machine code	Known as assembly code and also asm	There are many high level level languages e.g. c, java and they are called by their names.



Assembler	Compiler	Interpreter
Translate assembly code to machine code	Translate the entire high level language code to machine code	Translate the high level code line by line (single instruction at a time) and then convert to machine code

FETCH-EXECUTE-CYCLE

CPU executes the instruction with fetch-execute-cycle as;

Fetch

1. Fetch an instruction from memory.

2. Decode the instruction to determine the operation.

3. ·Fetch data from memory if necessary.

Execute

1. Perform the operation on the data.

2. Store the result in memory if needed.

REGISTERS

- ❑Records or collection of information
- ❑Storage area inside CPU, CPU take info, process and store it.
- ❑Fastest area present inside CPU
- ❑ Helps in Optimization of Processing time (HD file internally run on register and CPU process it)
- ❑Understanding of Hardware and Software interaction

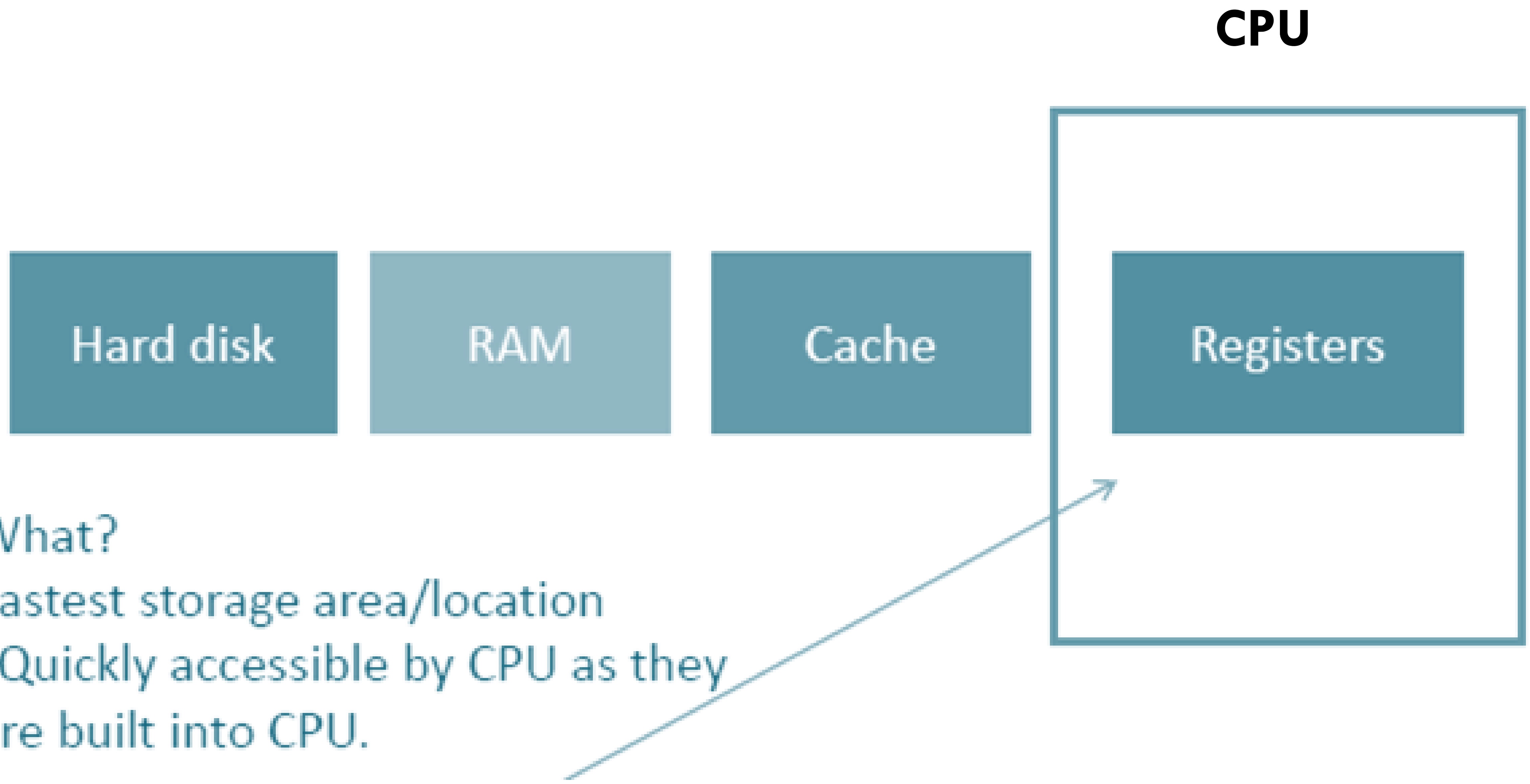
Processor operations mostly involve processing data. This data can be stored in memory and accessed from thereon. However, reading data from and storing data into memory slows down the processor, as it involves complicated processes of sending the data request across the control bus and into the memory storage unit and getting the data through the same channel.

To speed up the processor operations, the processor includes some internal memory storage locations, called **registers**.

The registers store data elements for processing without having to access the memory. A limited number of registers are built into the processor chip.

Registers are a type of computer memory used to quickly accept, store, and transfer data and instructions that are being used immediately by the CPU. The registers used by the CPU are often termed as Processor registers.

A processor register may hold an instruction, a storage address, or any data (such as bit sequence or individual characters).



CPU

Hard disk

RAM

Cache

Registers

What?

Fastest storage area/location

“Quickly accessible by CPU as they are built into CPU.

HD – Main storage- we run files present in HD.

HD file Run – RAM – Cache – Register

Register built in inside CPU, CPU quickly access information from register, program work processing time optimize.

Direct access register , time is saved , info save directly into Register because CPU extract info from here (CPU need space where it keep data to hold or remove).

CPU extract info from register through Assembly language programming.

Registers are the fastest memory locations built into microprocessor. Fast means CPU quickly access it close to CPU.

Chip by Intel 4004 in 1971, launched registered are used first time, by Federico Faggin

Following are the 14 types of registers