EEE304 – Digital Design with HDL (II)

Lecture 1

Dr. Ming Xu

Dept of Electrical & Electronic Engineering

XJTLU

In This Session

- Computer Architecture

Textbook:

Computer Organization and Design: The Hardware/Software Interface*, 3rd Ed.,* Patterson and Hennessy, Elsevier, 2005.

Assessment:

Final Exam (50%)

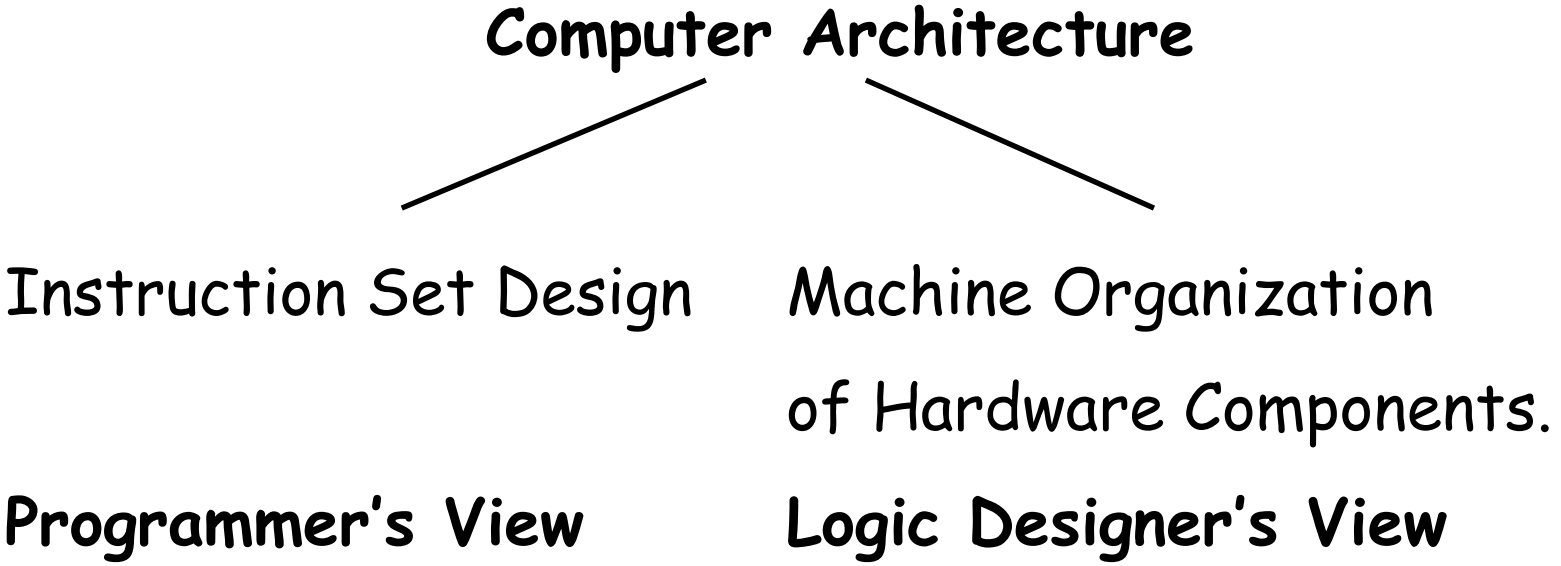
Assignment 1 (25%), Assignment 2 (25%)

Course Content

- Topics covered include :
  - Computer Architecture and Performance
  - Computer Instruction Sets
  - Processor Design
  - Pipelined Processor Design

What is Computer Architecture?

- Computer Architecture = Instruction Set Architecture (ISA, the interface between hardware and the lowest-level software) + Machine Organization



Language of the Machine

- Computer operates using number in base 2, or binary numbers
  - Why?
- Instructions are represented by binary numbers
- First programmers communicated using binary numbers
  - 100110010100000 represents the instruction to add two numbers together
- Programs invented to translate symbols to binary
  - Assembler
  - Less tedious and more readable
  - add A,B

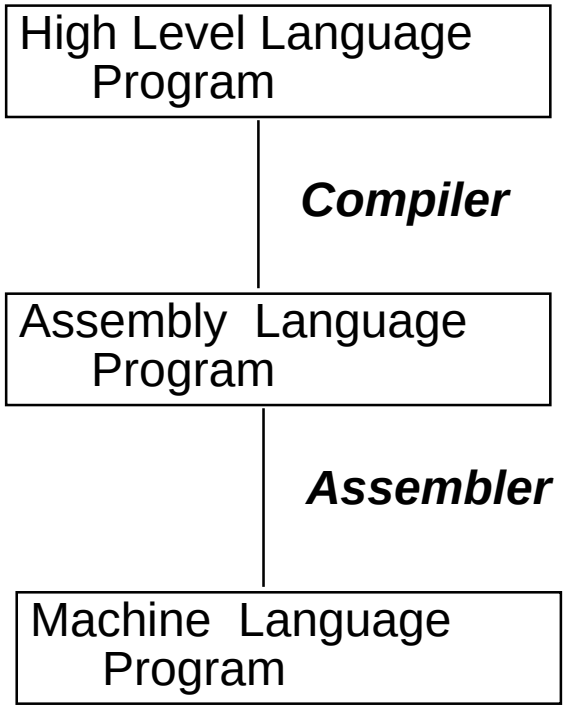
Language of the Machine

- Eventually, higher level languages were invented, which can be translated from higher level notations
  - Compiler
    - Program that accepts the more natural notation
  - Fortran was invented in 1954

Language of the Machine

- Benefits
  - Allow programmers to think in more natural language
  - Languages could be designed according to their intended use
    - Fortran for scientific computing
    - Cobol for business data processing
    - Lisp for symbol manipulation
  - Improve productivity of the programmers
  - Code portable between different platforms

Language of the Machine




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

```
lw  $15,  0($2)
lw  $16,  4($2)
sw $16,  0($2)
sw $15,  4($2)
```

```
0000 1001 1100 0110 1010 1111 0101 1000
1010 1111 0101 1000 0000 1001 1100 0110
1100 0110 1010 1111 0101 1000 0000 1001
0101 1000 0000 1001 1100 0110 1010 1111
```

Language of the Machine

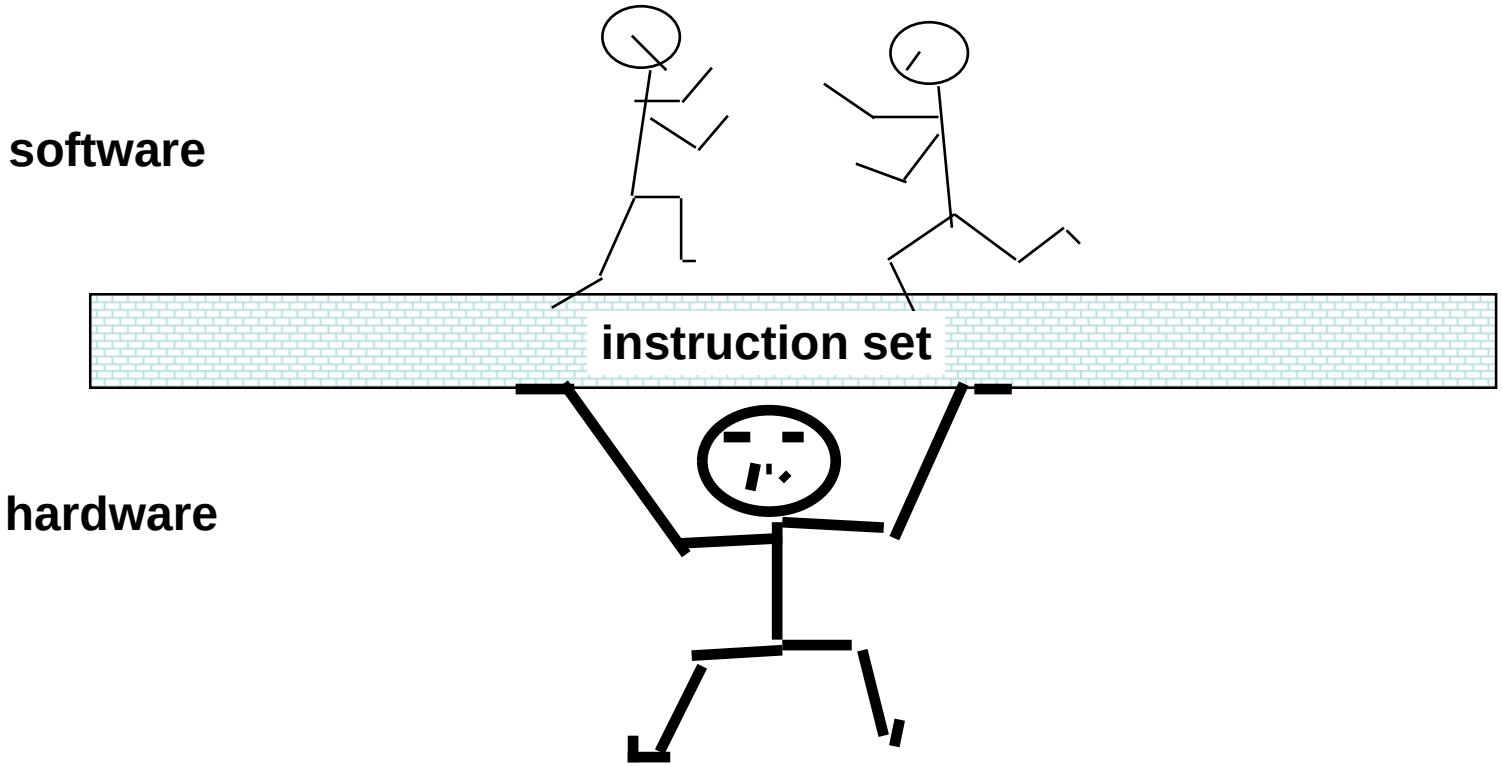
- Libraries of code developed
  - Reusing code more efficient than writing everything from scratch
- Operating systems were then developed to supervise the running of programs and allocate resources to the programs
- Categories of software
  - Systems software – aimed at programmers
    - OS, assembler, compiler
  - Application software – aimed at users
    - Spread sheets, text editors

Instruction Set Architecture

- Instruction set architecture has the attributes of a computing system as seen by the assembly language programmer or compiler. This includes
  - Instruction Set (what operations can be performed?)
  - Instruction Format (how are instructions specified?)
  - Data storage (where is data located?)
  - Addressing Modes (how is data accessed?)
  - Exceptional Conditions (what happens if something goes wrong?)

Instruction Set Architecture

- It is an abstraction interface between the hardware and the lowest level software
- Modern instruction set architectures: 80x86/Pentium, PowerPC, DEC Alpha, MIPS, SPARC, HP



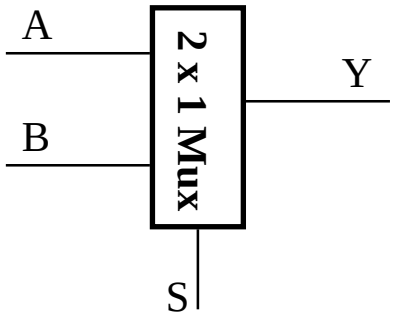
Abstraction

- An Abstraction has **levels**. Delving into the depths of these levels reveals more information.
- An abstraction omits unneeded detail, helps us cope with complexity.
- Abstraction is the approach used in the design of software and hardware.
- An **abstraction system** consists of hierarchical levels with each lower level hiding details from the level above.

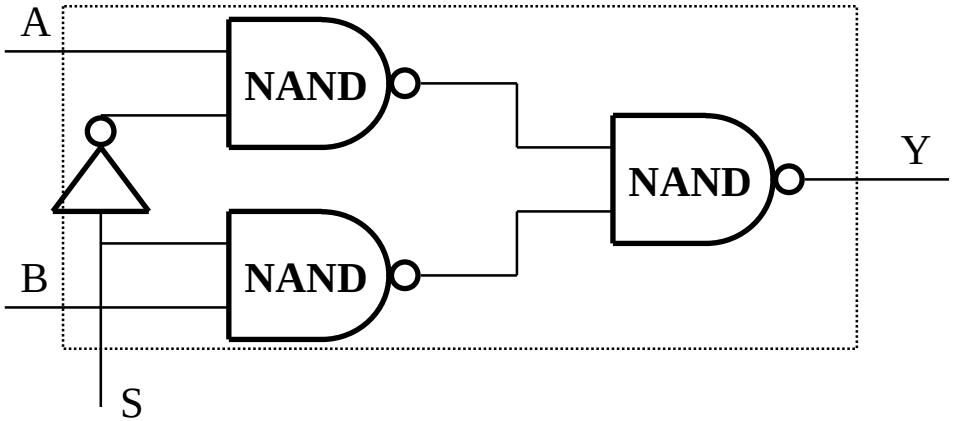
Levels of abstraction

- Computer architecture uses various levels of abstractions.
- Each level of abstraction consists of
  - an **interface** (outside view of what it does), and
  - an **implementation** (inside view of how it works)

Interface

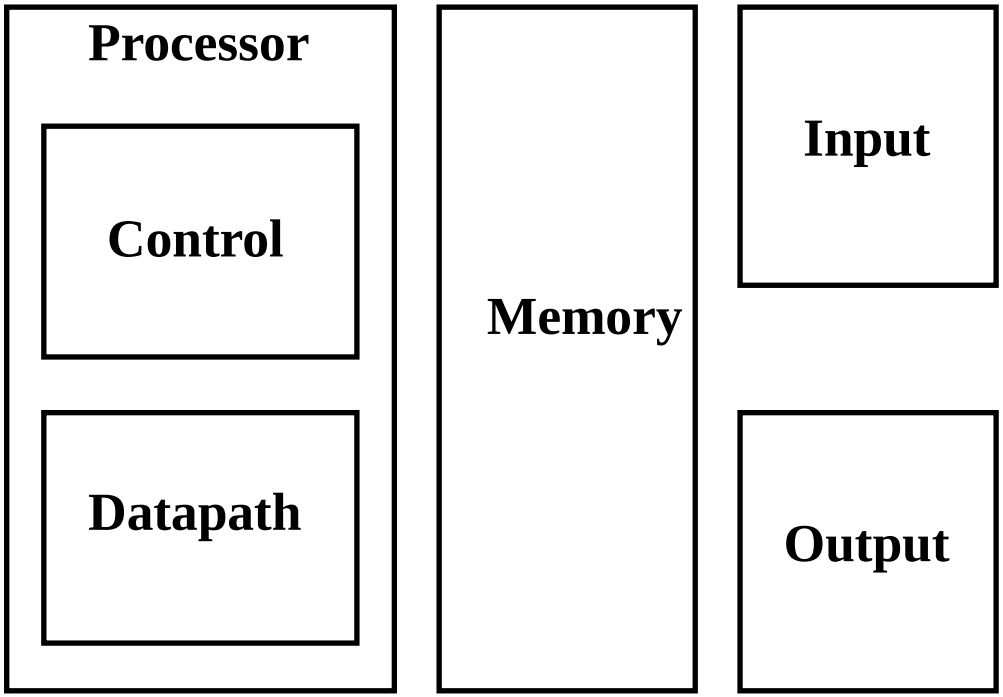


Implementation



The Organization of a Computer

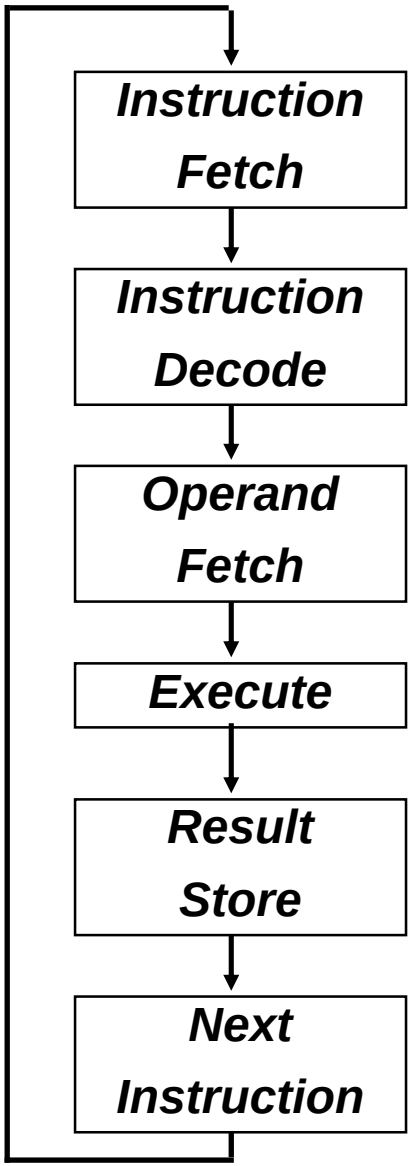
- Since 1946 all computers have had 5 main components



The Organization of a Computer

- **Processor**:
  - **Data path** - to do arithmetic and logic: e.g. adders, multipliers, shifters
  - **Control** - to give directions to the other components: e.g. tells the data path, memory, and IO devices what to do according to the instructions of the program
- **Memory:** Holds data and instructions**: e**.g. cache, main memory, disk.
- **Input:** Sends data to the computer: e.g. keyboard, mouse.
- **Output:** Gets data from the computer: e.g. screen, sound card, printer.

Instruction Execution



Obtain instruction from program storage

Determine required actions and instruction size

Locate and obtain operand data

Compute result value or status

Deposit results in storage for later use

Determine successor instruction