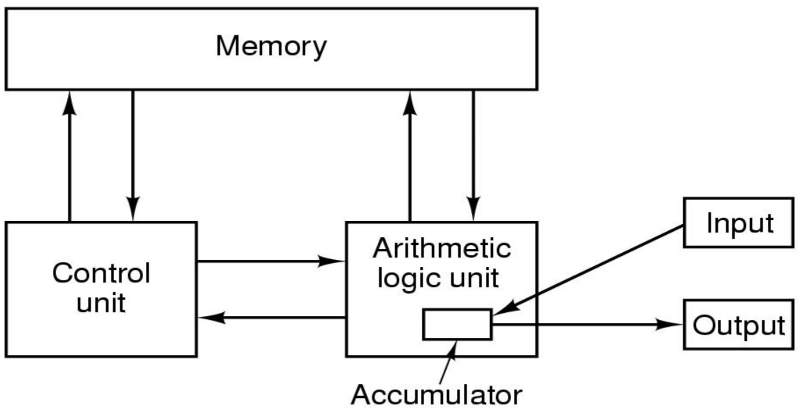
Logic Design

Abstraction Levels

The von Neumann Architecture

- All general purpose computers are now based on the key concepts of the von Neumann architecture:
  - A single read-write memory for data and instructions
  - The memory is addressable by location in a way which does not depend on the contents of the location
  - Execution proceeds using instructions from consecutive locations unless an instruction modifies this sequentiality explicitly
- The von Neumann bottleneck: a single channel for both instructions and data
- Harvard Architecture: separate memories - double the bandwidth of the simple von Neumann architecture

The von Neumann and Harvard Architectures



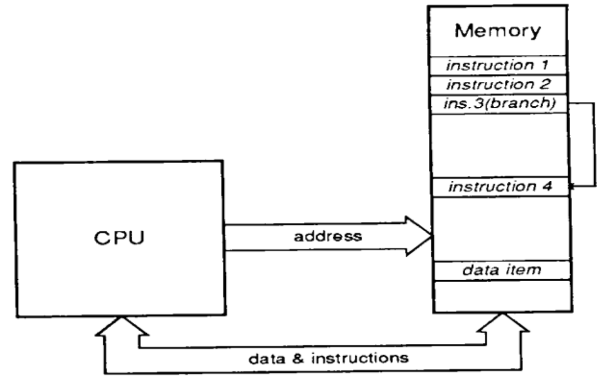


Figure 2: The von Neumann architecture

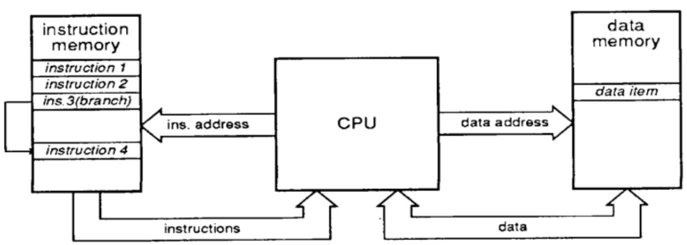


Figure 3: The Harvard architecture

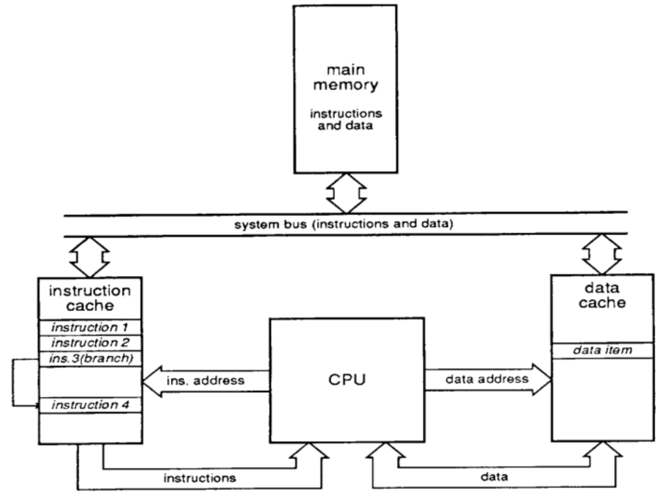


Figure 4: A modified Harvard architecture

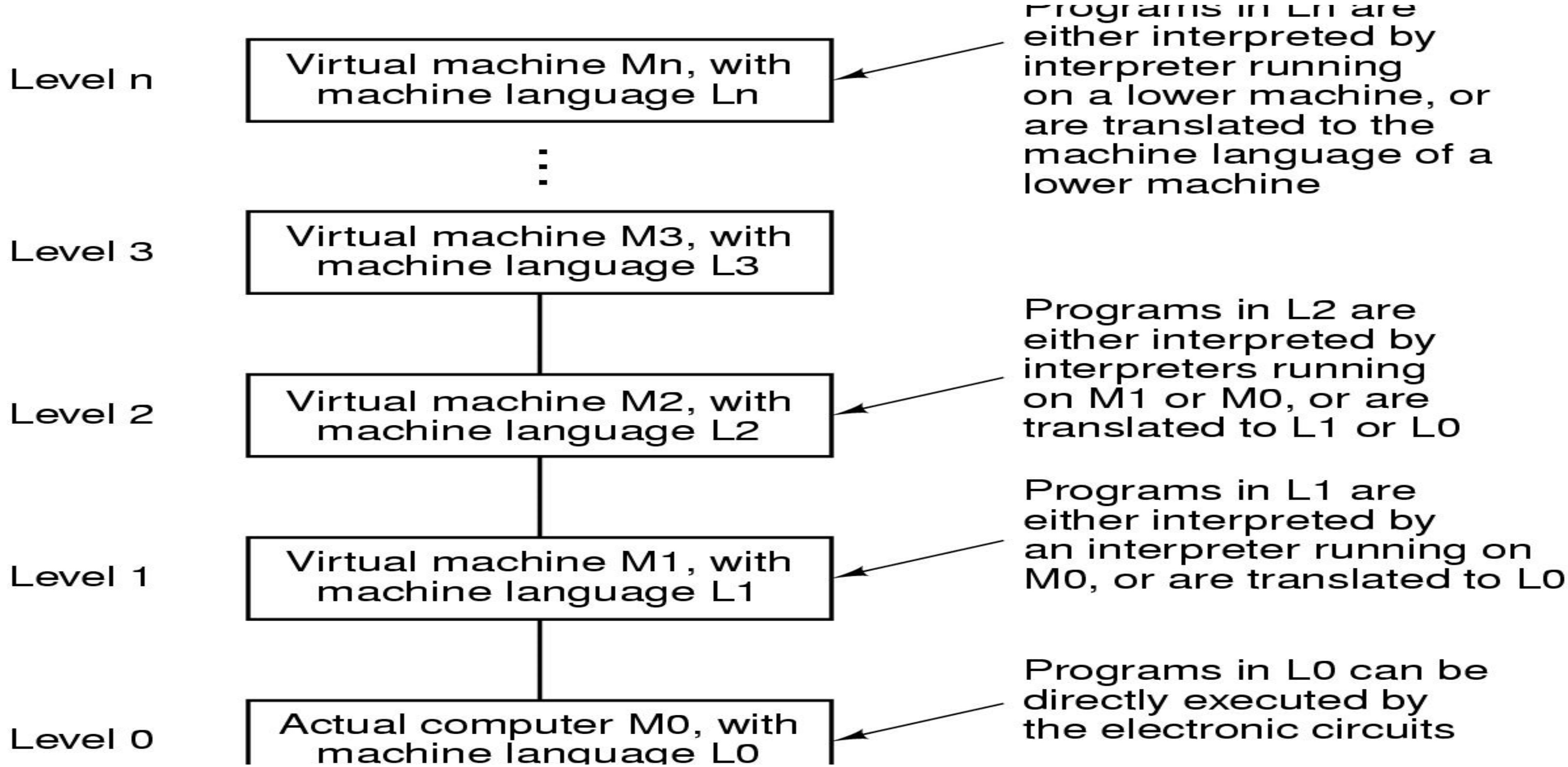
Computer Components

- CPU
  - Datapath
  - Control
- Memory (hierarchy)
  - Main Memory
  - Secondary memory
  - Cache
- I/O devices
- Buses (external and internal to CPU)

Instruction Execution: The Fetch-Decode-Execute Cycles

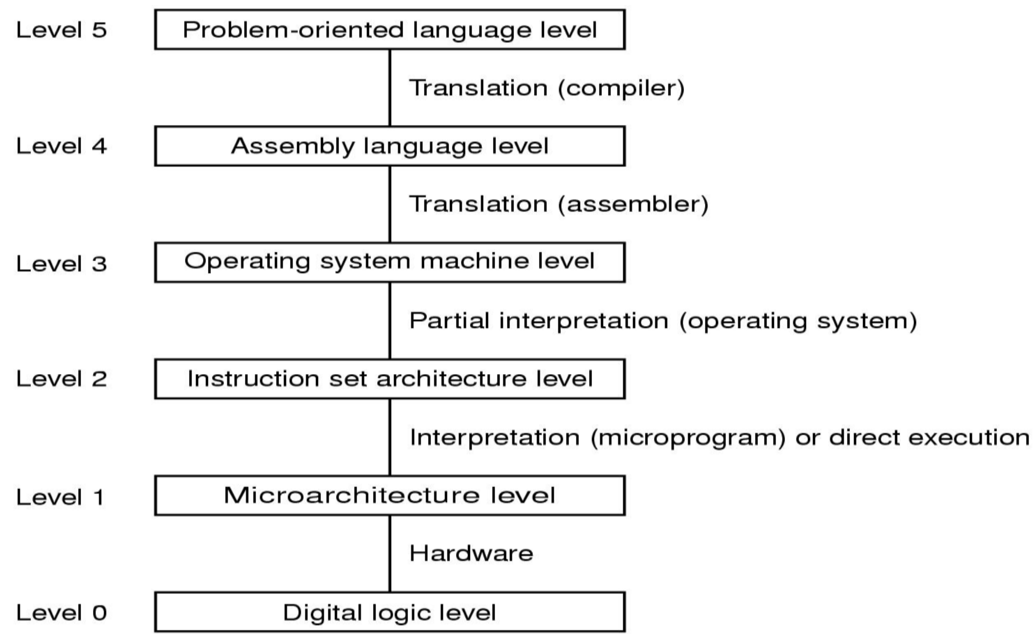
- The CPU executes instructions in a series of small steps:
  - **Fetch** the next instruction from memory in the *Instruction Register (IR)*
  - Change the *Program Counter (PC)* to point to the following instruction
  - **Determine the type** of fetched instruction
  - If the instruction uses a word in memory, determine where it is
  - Fetch the word, if needed, into the CPU
  - **Execute** the instruction
  - Go to step 1

Multi-Level Approach



Contemporary Typical Multilevel Machines

- Digital Logic design. Gate level
- Micro-architecture Level
  - Programming Model (Registers)
  - Datapath & Control
- Instruction Set Architecture Level-ISA
  - Instruction types and formats
  - Addressing



Translation Example

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;
}
```

C compiler

Assembly language program (for MIPS)

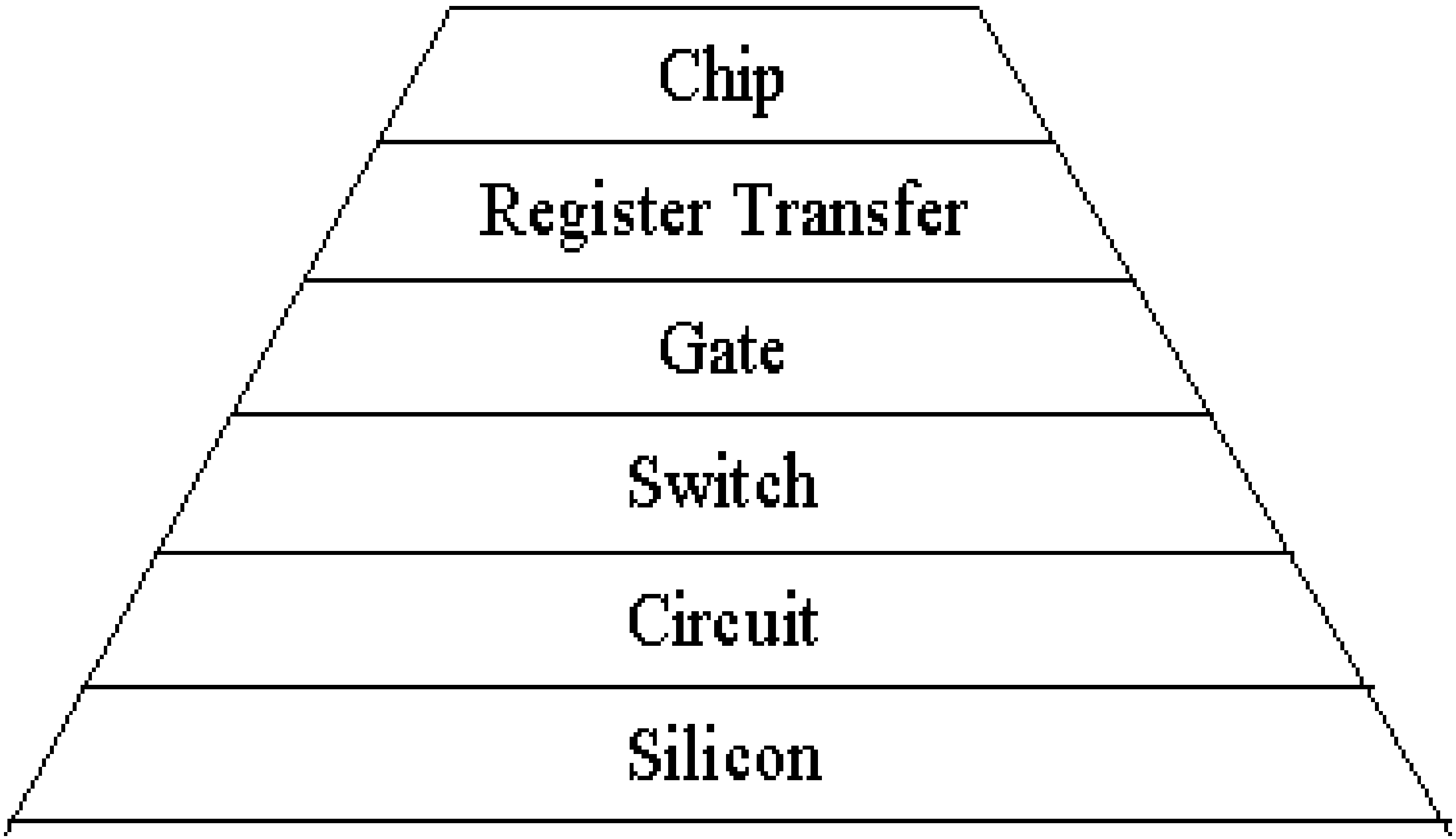
```
swap:
      muli $2, $5,4
      add  $2, $4,$2
      lw   $15, 0($2)
      lw   $16, 4($2)
      sw   $16, 0($2)
      sw   $15, 4($2)
      jr   $31
```

Assembler

Binary machine language program (for MIPS)

```
00000000101000010000000000011000
00000000100011100001100000100001
10001100011000100000000000000000
10001100111100100000000000000100
10101100111100100000000000000000
10101100011000100000000000000100
00000011111000000000000000001000
```

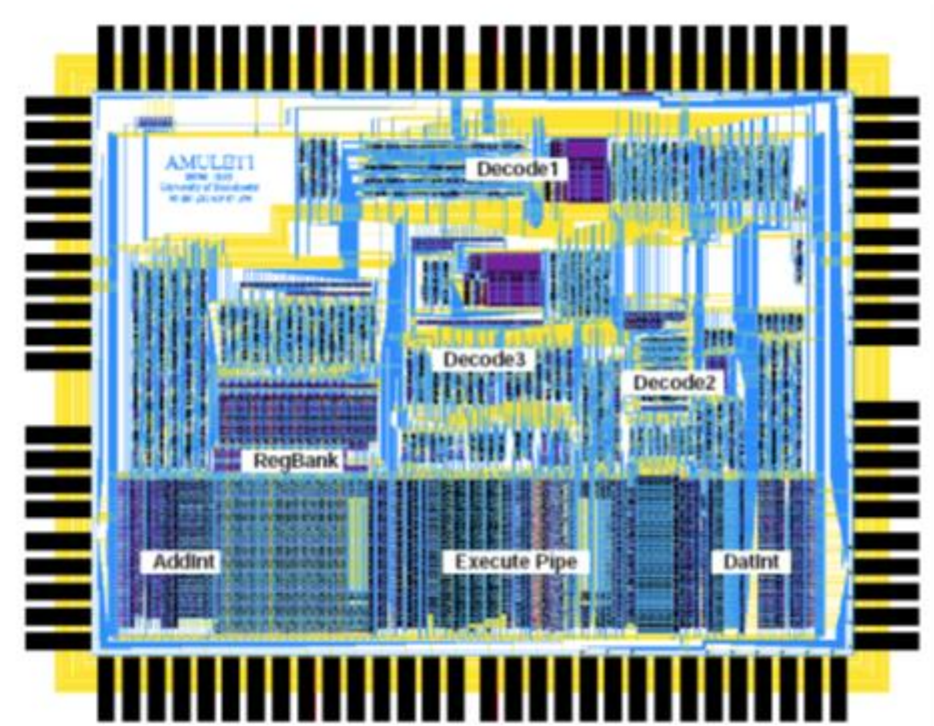
Computer Architecture Description Levels

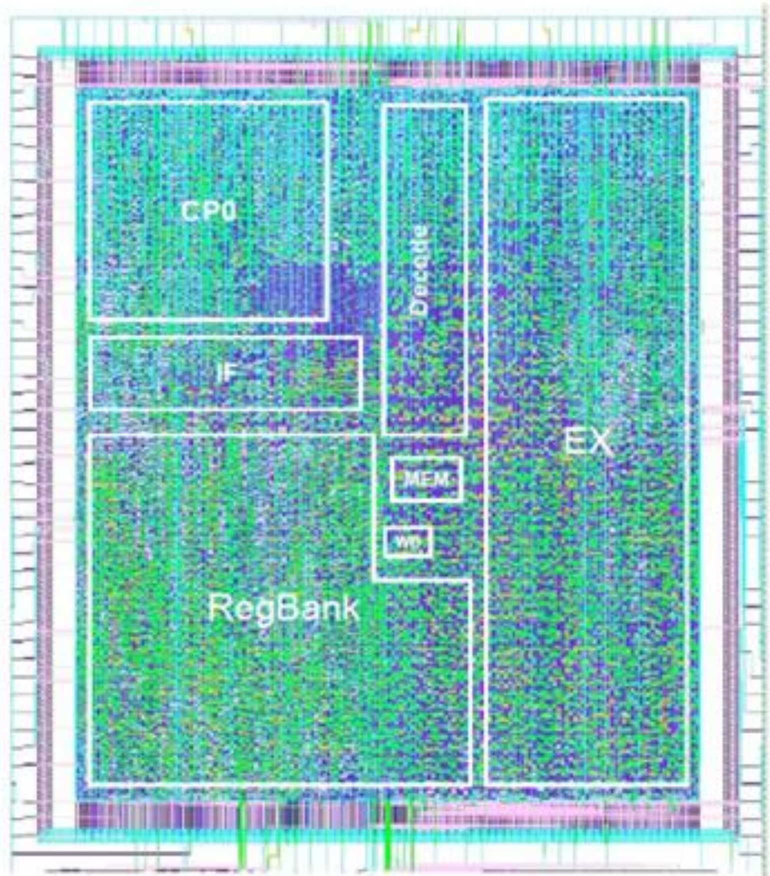


Silicon Level

- The real *geometry* of the physical *layout* of materials such as diffusion, polysilicon, and metals on the silicon surface.
- Schematic, graphical or textual layout representation.
- The layout is a symbolic, non-mathematical model, a purely morphological description, without any behavioural information incorporated.

Example Silicon Level Descriptions





Circuit Level

- System expressed in terms of traditional passive and active *electrical circuit components* (e.g resistors, capacitors and transistors).
- Circuit specified as a schematic diagram or via a textual description (*net list)*. The circuit net list may be specified explicitly or be extracted from the physical layout or circuit schematics.
- Input and output of each component in the model have analogue values. The analogue behavior of the components at this level is typically expressed in terms of *differential equations* which may be solved by a circuit simulator (e.g. SPICE)
- Circuit simulators typically use as input the circuit net list.

Switch Level

- System described at the same level of abstraction as in the circuit level, however only digital, rather than analogue, *signal values* are considered: {1,0}, {on,off}, {high, low}.
- The same circuit diagram and circuit net list are used, but components are modelled in a much simpler way: Instead of a precise analogue behaviour *only the digital behaviour* is described.
- Transistors are modelled as switches, with two states, ``on'' (low impedance) and ``off'' (high impedance). A model at switch level, and all levels above that, may be simulated using a *discrete event simulator*.

Gate Level

- The ***logic design*** level, where the implementation of the system in terms of gates (AND, OR, inverters etc) and flipflops is described.
- Description provided graphically, by means of a logic diagram, or textually, using a *Hardware Description Language*
- A gate level net list may be automatically extracted from the schematic or textual specification. The net list may be provided as input to a *discrete event logic simulator*.
- Gate level has traditionally been the main design level for digital systems.

Register Transfer Level (RTL)

- The system is expressed in terms of higher level components such as registers, counters, multiplexers, ALUs, multipliers, shifters, memory blocks etc. (*functional blocks*, *functional level modelling*).
- RTL components are primitive behavioural models directly expressed using a functional HDL (In principle, though, RTL components may be expressed in terms of an interconnection of lower level primitives e.g gates)
- RTL models may be simulated by a functional *discrete event simulator*.

Chip (Architectural) Level

- The *structural primitives* of the model are blocks such as processors, memories, serial and parallel ports, interrupt controllers etc.
- Typically, the model boundaries are defined by the chip boundaries, however this requirement is not restrictive (e.g. when modelling parallel architectures, where the system consists of more that one chip).
- As with the functional blocks in RTL, chip level components are primitive behavioural models and are not specified hierarchically in terms of lower level blocks.
- *Discrete event simulation* is employed for the execution of chip level models.