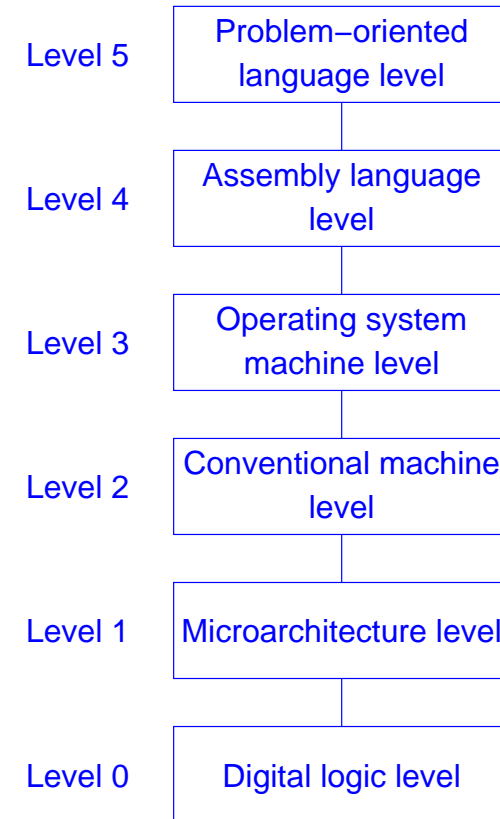


Modern Computer Organization: Abstractions at Several Levels

- levels of computer architecture
- the Java Virtual Machine (level 5 - mainly)
- memory hierarchy and caches (level 1)
 - issues, and details
- the digital logic level (level 0)
- revision from lectures D1–C3

Levels of Computer Architecture

- Level 5: Problem-oriented language (e.g. Java, C?)
 - compilation
- Level 4: Assembly language
 - translation
- Level 3: Operating system machine
 - partial interpretation
- Level 2: Instruction set architecture
 - interpretation (microprogram) or direct execution
 - e.g. registers, (virtual) memory access
- Level 1: Microarchitecture
 - hardware
 - e.g. datapaths, buses, ALUs, 'hidden' registers (CI, MAR),
- Level 0: Digital logic
 - e.g. circuits (wires and gates)



alt. [Null&Lobur, fig 1.3]
– note the extra level!

High-Level Language Level: the Java Virtual Machine (JVM)

Ref: [Null&Lobur, sect 3.5]

Fig 8.11, Fig 8.13

- JVM is a stack machine interpreter for Java bytecode (`.class` files)
- individual bytecode instructions are represented by numbers stored in a single byte
 - the Java code sequence `i = i + 1;` (where `int i;`) might translate into the bytecode sequence:

mnemonic	bytecode	operation
<code>iload_0</code>	1A	push local variable 0 (i) onto stack
<code>iconst_1</code>	04	push constant 0 onto stack
<code>iadd</code>	60	pop the top two entries then push their sum on the stack
<code>istore_0</code>	3A	pop the stack and store value in local variable 0

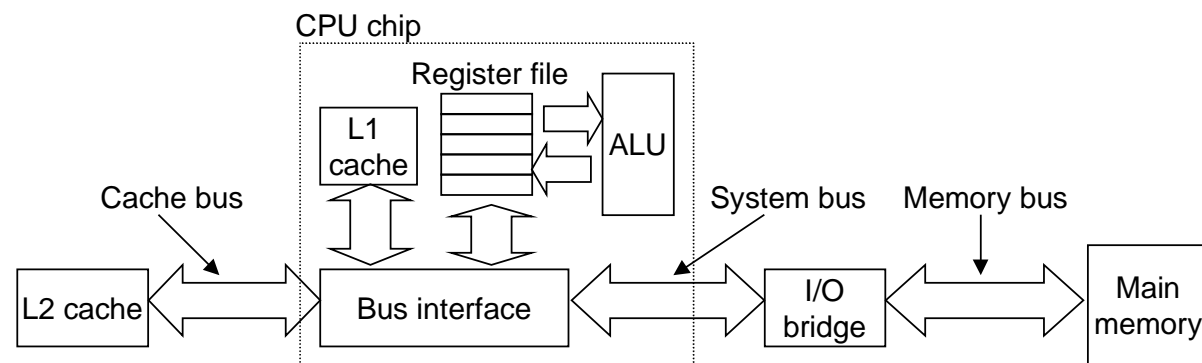
- a JVM *interprets* these, and can produce the same effects on many different physical computers
 - this stage corresponds to Level 2, but on a *virtual* (software) level
- OS level is abstracted via I/O-related methods (accessed via the JVM native methods, which accesses the C system calls)
- Q: why is it useful to design Java this way? why use a stack machine?

The Microarchitectural Level Example: the Memory Hierarchy

Ref: [Null&Lobur, sect 6.3-6.4], [O'H&Bryant, sect 6.2-6.4]

- in a computer system, there is a memory hierarchy, because:
 - many different mediums for the storage of data
 - generally, there is a *trade-off* between speed and capacity
 - ◆ fast memories tend to be small; large memories tend to be slow

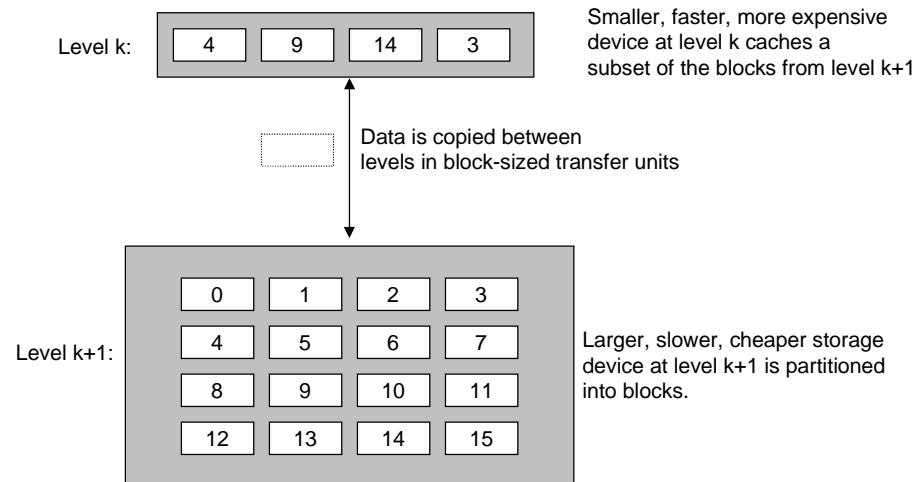
medium	access time	typical size
registers	~1 ns	< 1 KB
cache mem.	~ 20 ns	< 2 MB
main mem.	~ 100 ns	< 2 GB
disk	~ 10 ⁷ ns	> 10 GB



[O'H&Bryant, fig 6.24]

- idea of cache memory: store recently accessed data from main memory in a small, faster memory (closer to / inside the CPU)
 - Q: why might this be useful?

Cache Memory: Issues

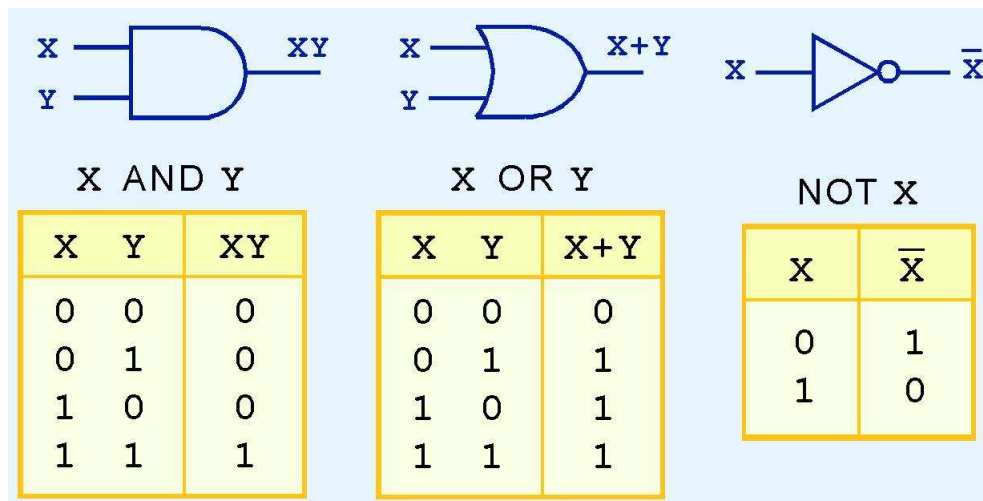


- [O'H&Bryant, fig 6.22]
- issue: cache must store the memory address (as a 'tag') of the data, as well as the data itself!
- issue: (considering the cache is to be organized as an array) which cache entry ('index') holds the data for address X (if any)?
- idea (direct-mapped cache): for a cache of size $C' = 2^c$ entries, all addresses with the same a_1 are mapped to the same entry (does all of X need be in the tag?)

$$X = \begin{array}{|c|c|} \hline 31 & c \quad c-1 \quad 0 \\ \hline a_0 & a_1 \\ \hline \end{array}$$

The Digital Logic Level: Gates

- Ref: [Null&Lobur, sect 2.3–2.4], [Tanenbaum, ch 3]
- digital logic is lowest level of computer organization!
- digital circuit elements (including 1-bit memory cells) are made from gates
(in turn, from 1 or more transistors or switches, [Tanenbaum, fig 3.1])
- [Null&Lobur, figs 3.1-2]: AND and OR gates manipulate voltage levels (0- low, 1 - high) according to truth tables (Boolean logic)

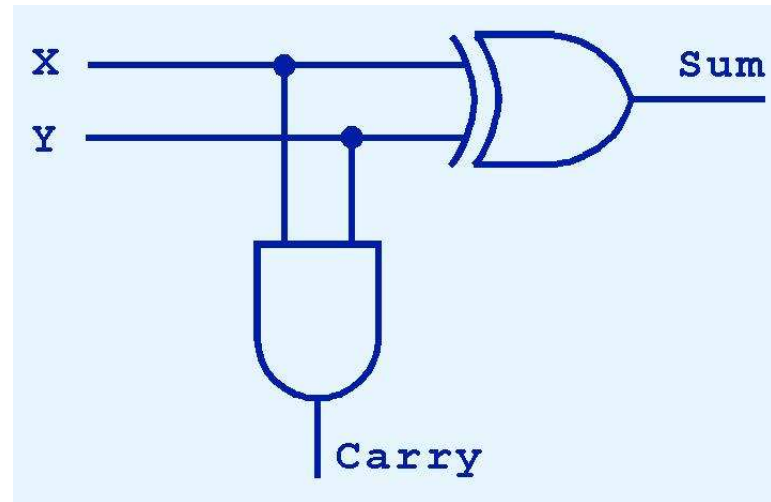


- [O'H&Bryant, p 45]: set of colors forming an 8-element boolean algebra

The Digital Logic Level: Half Adders

- from gates, higher-level components such as a half-adder are constructed ([Null&Lobur, figs 3.10,3.11]: $\text{Sum} = X \text{ XOR } Y$, $\text{Carry} = X \text{ AND } Y$)

Inputs		Outputs	
X	Y	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



- this can be extended to a full adder ([Null&Lobur, fig 3.13])
(truth table, circuit)
 - how could this be used to implement an integer add circuit?
- bits can be stored in components with feedback circuits (with clocked inputs)
 - e.g. SR latch, [Null&Lobur, fig 3.21]
- discussion point: why is binary the representation used in computers?

Revision from Lectures D1–D3, C1–C3

winners from Minute Paper 1 are:

- big and little endian (D2)
- floating point issues
 - Q: 32-bit IEEE floating point has 8 bits for the exponent and 23 bits for the mantissa. If this were changed to 10 bits for the exponent and 21 bits for the mantissa:
 - (a) it would make no real difference
 - (b) you would have greater range but less precision
 - (c) you would have greater precision but less range
 - conversions
- instruction sets: RISC vs CISC
- pointers in C
- dynamically allocated arrays