CS107, Lecture 15
Accessing the Architecture: An Introduction to Comp Arch

Reading: B&O 3.1-3.4

This document is copyright (C) Stanford Computer Science, Adam Keppler and Joel Ramirez, licensed under Creative Commons Attribution 2.5 License. All rights reserved.
Based on slides created by Nick Troccoli, Chris Gregg, and Raymond Klefstad

What should someone do if they find a vulnerability? How can we incentivize responsible disclosure?

Disclosure

What's the best way to disclose vulnerabilities?

- **Full disclosure?** Make vulnerabilities public as soon as they are found? *Few people now endorse this approach due to its drawbacks.*
- **Responsible disclosure?** Privately alert software maker to fix in reasonable amount of time before publicizing? *Most common, and recommended by ACM code of ethics.*

Disclosure

- Various entities may want to financially reward people for finding and reporting vulnerabilities.
- The US Federal Government is one of the largest discoverers and purchasers of 0-day vulnerabilities. It follows a “Vulnerability Equities Process” (VEP) to determine which vulnerabilities to responsibly disclose and which to keep secret and use for espionage or intelligence gathering.

How do we weigh competing stakeholder interests here, such as country vs. individual?

Partiality

Partiality holds that it is acceptable to give preferential treatment to some people based on our relationships to them or shared group membership with them.

Impartiality, involves "acting from a position that acknowledges that all persons are ... equally entitled to fundamental conditions of well-being and respect."

Partiality

Degrees of Partiality

Partiality: preference towards own family, friends, and state is morally acceptable or even required

Partial Cosmpolitanism: limited preference towards own state acceptable

Universal Care: preference towards family acceptable but not towards state

Impartial Benevolence: same moral responsibilities towards all people

Case Study: EternalBlue

2012-2017: NSA secretly stores the EternalBlue Microsoft vulnerability and uses it to spy on both US and non-US citizens.

early 2017: EternalBlue stolen by hacker group the ShadowBrokers. NSA discloses EternalBlue to Microsoft.

March 14, 2017: Microsoft releases a patch for the vulnerability.

May 12, 2017: EternalBlue is the basis of the WannaCry and other ransomware attacks, leading to downtime in critical hospital and city systems and over $1 billion of damages.

Microsoft's Argument

"[T]his attack provides yet another example of why the **stockpiling of vulnerabilities** by governments is such a problem. ...

We need governments to consider the **damage to civilians** that comes from hoarding these vulnerabilities and the use of these exploits.

This is one reason we called in February for a new "Digital Geneva Convention" to govern these issues, including a **new requirement for governments to report vulnerabilities to vendors**, rather than stockpile, sell, or exploit them.

And it's why we've pledged our support for **defending every customer everywher**e in the face of cyberattacks**, regardless of their nationality**."

Full post here

Critical Questions

- Do we have special obligations to our own country and to protect our people? If so, what would this mean?
- If intentionally exploiting a vulnerability is wrong when done by a private citizen, is it equally wrong when done by the government?
- Should I be loyal to my country, a citizen of the world, or both?
- When should I give preference to my family members and when should I strive to treat all equally?

What you choose matters – the moral obligations you take on constitute who you are.

Revisiting EternalBlue

Federal Government

Microsoft

Partiality: preference towards own family, friends, and state is morally acceptable or even required	Partial Cosmpolitanism: limited preference towards own state acceptable	Universal Care: preference towards family acceptable but not towards state	Impartial Benevolence: same moral responsibilities towards all people

Partiality Takeaways

- Understanding partiality helps us understand how we balance cases of competing interests and where we may personally fall on this spectrum.
- In order to evaluate situations, it's critical to understand the good and the bad that may come of it (e.g. EternalBlue). Better understanding privacy and privacy concerns is critical to this! (more later)

GCC Optimizations

Tail Recursion

Tail recursion is an example of where GCC can identify recursive patterns that can be more efficiently implemented iteratively.

```
long factorial(int n) {
  if (n <= 1) {
      return 1;
  }
  else return n * factorial(n - 1);
}
```

Tail Recursion Example

Recall the factorial problem from assembly lectures:

```
unsigned int factorial(unsigned int n) {
    if (n <= 1) {
        return 1;
    }
    return n * factorial(n - 1);
}
```

What happens with **`factorial(-1)`**?

- Infinite recursion → Literal stack overflow!
- Compiled with `-Og`!

Factorial: -Og vs –O2

```
401146 <+0>:  cmp    $0x1,%edi
401149 <+3>:  jbe    0x40115b <factorial+21>
40114b <+5>:  push   %rbx
40114c <+6>:  mov    %edi,%ebx
40114e <+8>:  lea    -0x1(%rdi),%edi
401151 <+11>:callq  0x401146 <factorial>
401156 <+16>:imul   %ebx,%eax
401159 <+19>:pop    %rbx
40115a <+20>:retq
40115b <+21>:mov    $0x1,%eax
401160 <+26>:retq
```

-O2:

- What happened?
- Did the compiler “fix” the infinite recursion?

```
4011e0 <+0>:  mov    $0x1,%eax
4011e5 <+5>:  cmp    $0x1,%edi
4011e8 <+8>:  jbe    0x4011fd <factorial+29>
4011ea <+10>:nopw   0x0(%rax,%rax,1)
4011f0 <+16>:mov    %edi,%edx
4011f2 <+18>:sub    $0x1,%edi
4011f5 <+21>:imul   %edx,%eax
4011f8 <+24>:cmp    $0x1,%edi
4011fb <+27>:jne    0x4011f0 <factorial+16>
4011fd <+29>:retq
```

Breaking Down the –O2

```
4011e0 <+0>:  mov    $0x1,%eax                    # Initialize %eax with 1.
4011e5 <+5>:  cmp    $0x1,%edi                    # Compare input value (%edi) with 1.
4011e8 <+8>:  jbe    0x4011fd <factorial+29>      # If input <= 1 (unsigned check), jump to return.
4011ea <+10>: nopw   0x0(%rax,%rax,1)             # No operation (probably for alignment).
4011f0 <+16>: mov    %edi,%edx                    # Copy current value of %edi to %edx.
4011f2 <+18>: sub    $0x1,%edi                    # Decrement %edi.
4011f5 <+21>: imul   %edx,%eax                    # Multiply %eax by %edx and store result in %eax.
4011f8 <+24>: cmp    $0x1,%edi                    # Compare decremented value of %edi with 1.
4011fb <+27>: jne    0x4011f0 <factorial+16>      # If %edi is not 1, repeat the multiplication.
4011fd <+29>: retq                                # Return with the result in %eax.
```

-02:

- Recursive -> Iterative
- No Stack Overflow, Saves Memory and Operations

GCC Optimizations

- Constant Folding
- Common Sub-expression Elimination
- Dead Code
- Strength Reduction
- Code Motion
- Tail Recursion
- **Loop Unrolling**

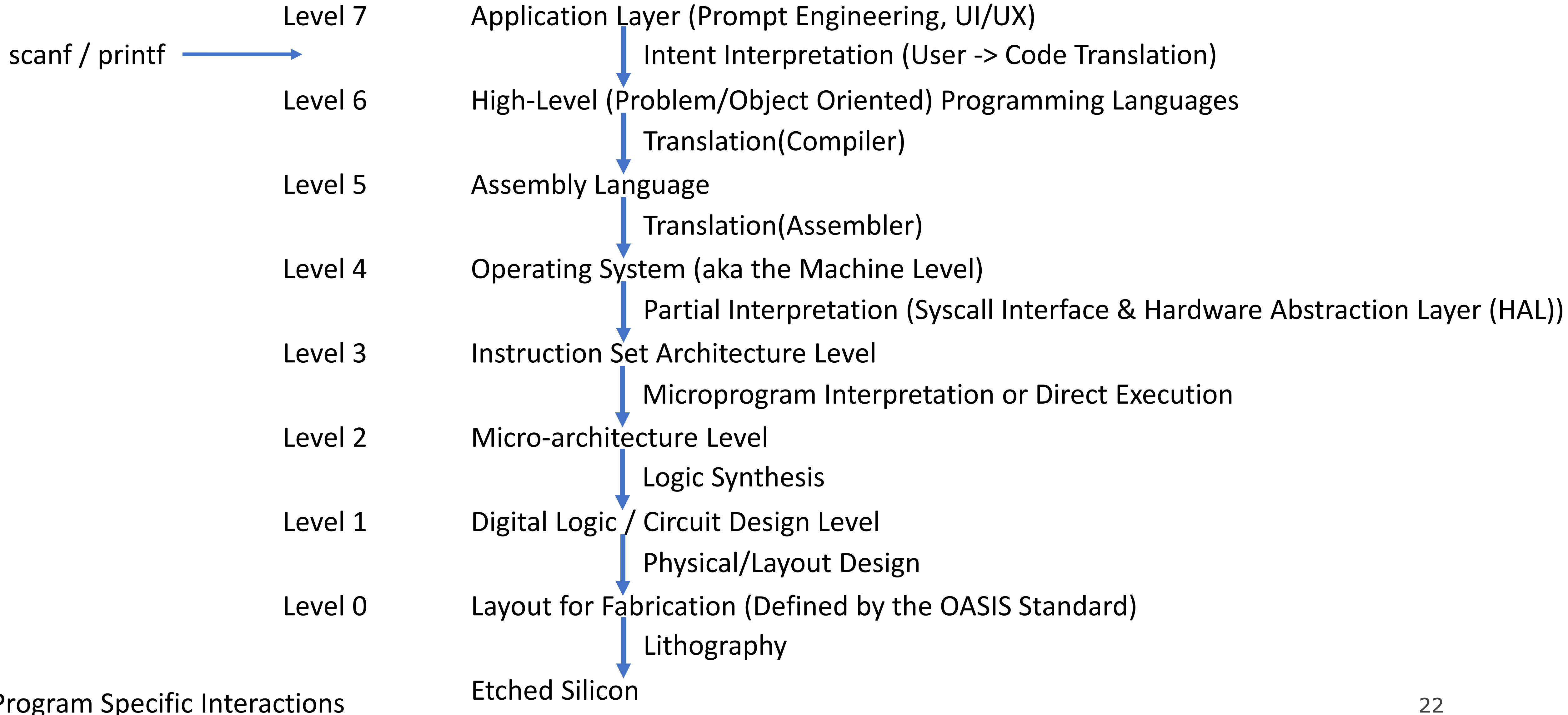
Loop Unrolling

Loop Unrolling: Do **n** loop iterations' worth of work per actual loop iteration, so we save ourselves from doing the loop overhead (test and jump) every time, and instead incur overhead only every n-th time.

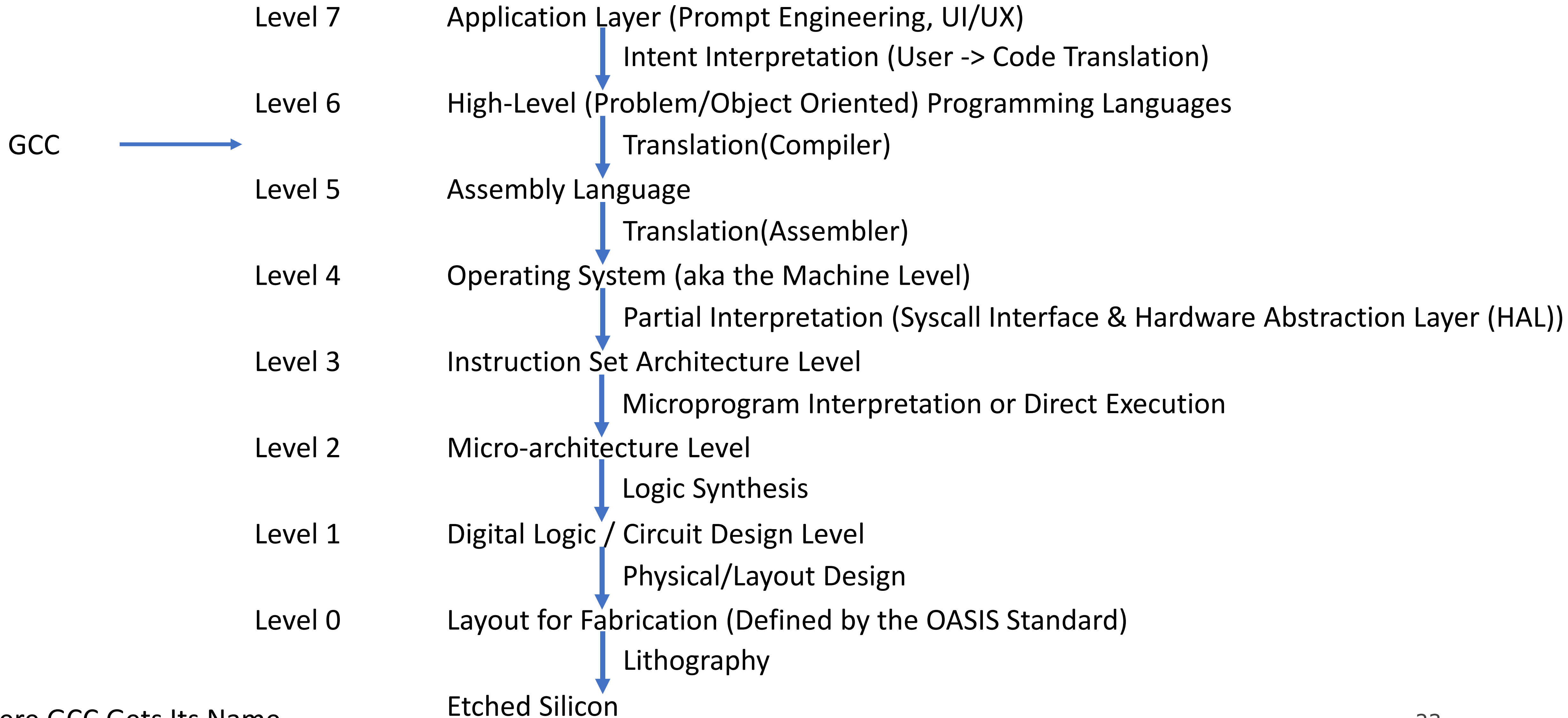
```
for (int i = 0; i <= n - 4; i += 4) {
     sum += arr[i];
     sum += arr[i + 1];
     sum += arr[i + 2];
     sum += arr[i + 3];
} // after the loop handle any leftovers
```

Into the Architecture!

Programming Levels

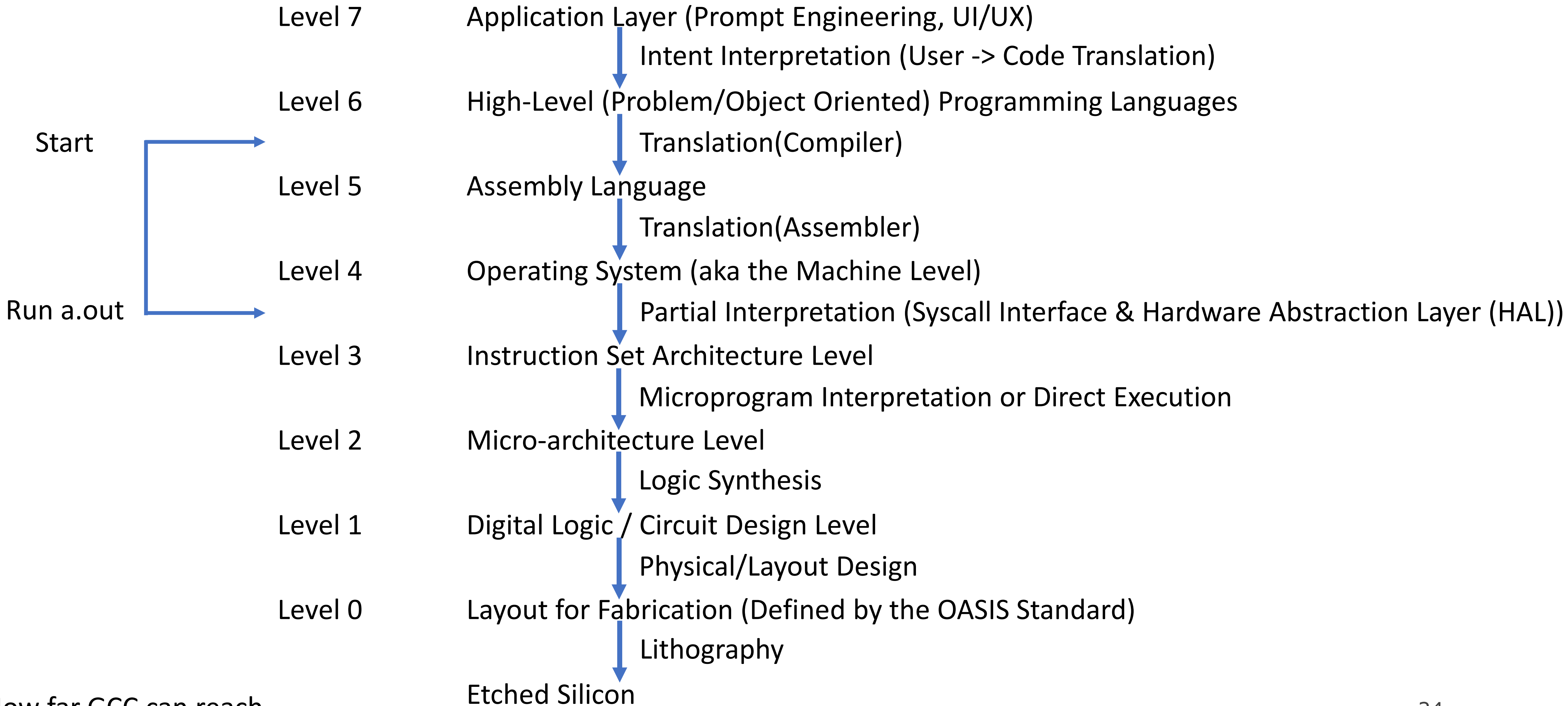


Programming Levels

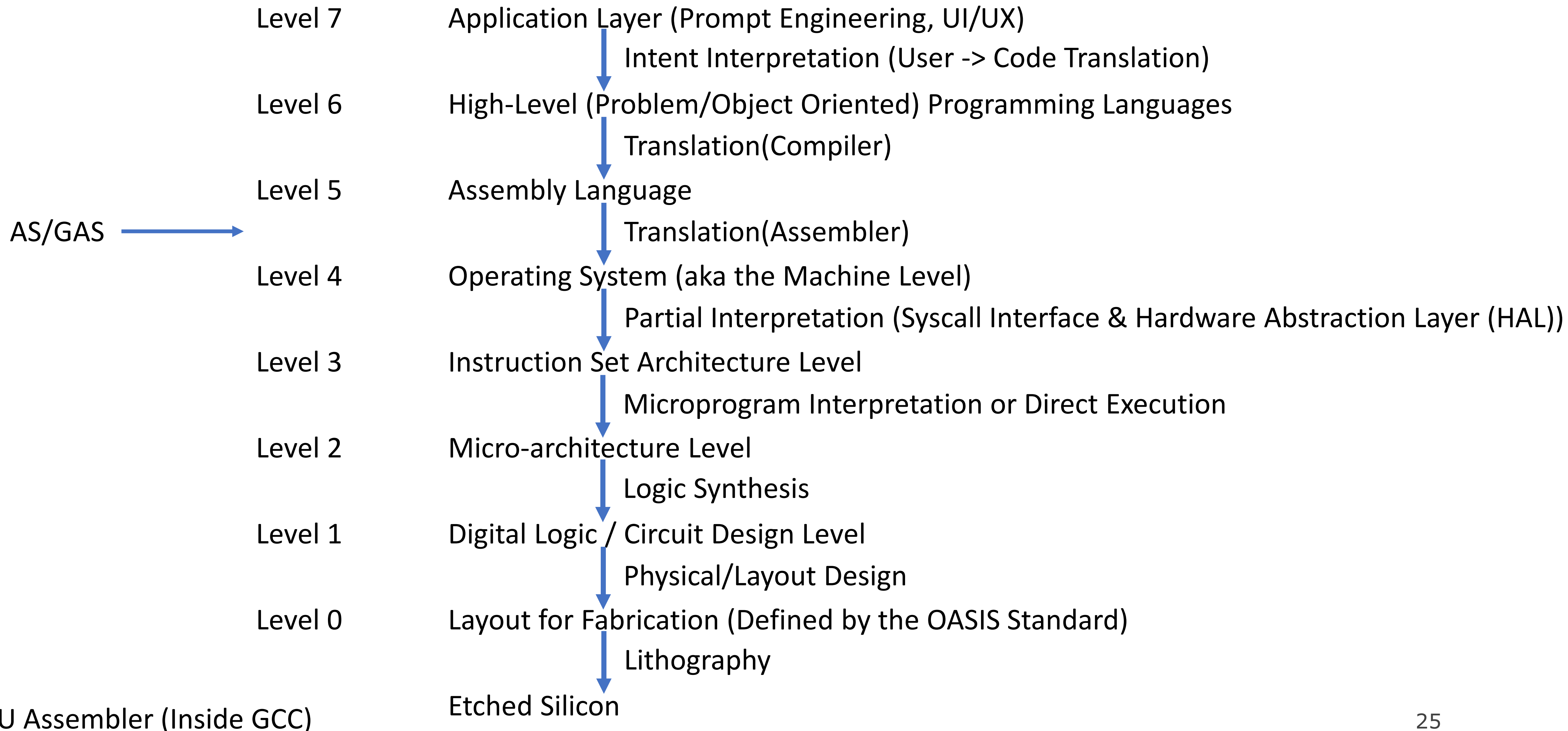


Where GCC Gets Its Name

Programming Levels

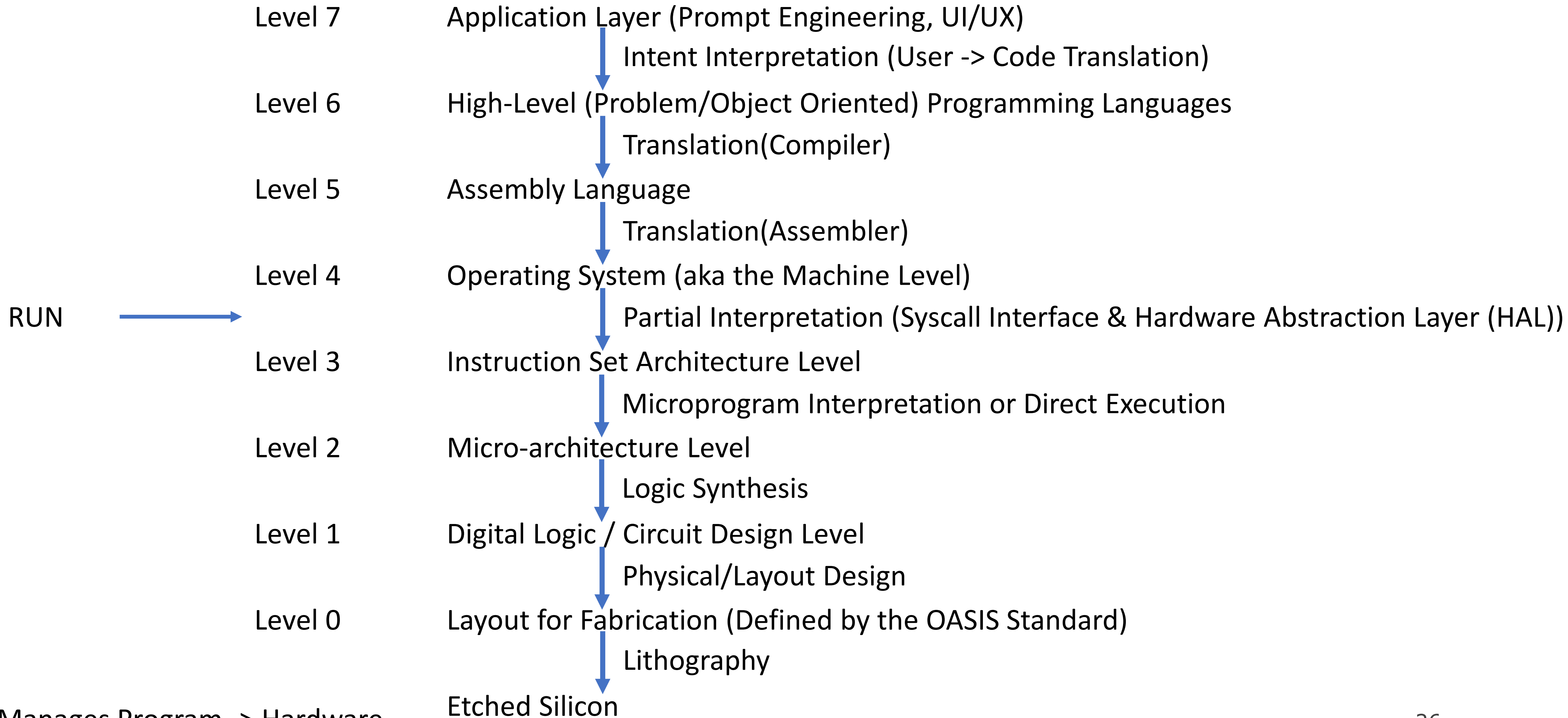


Programming Levels

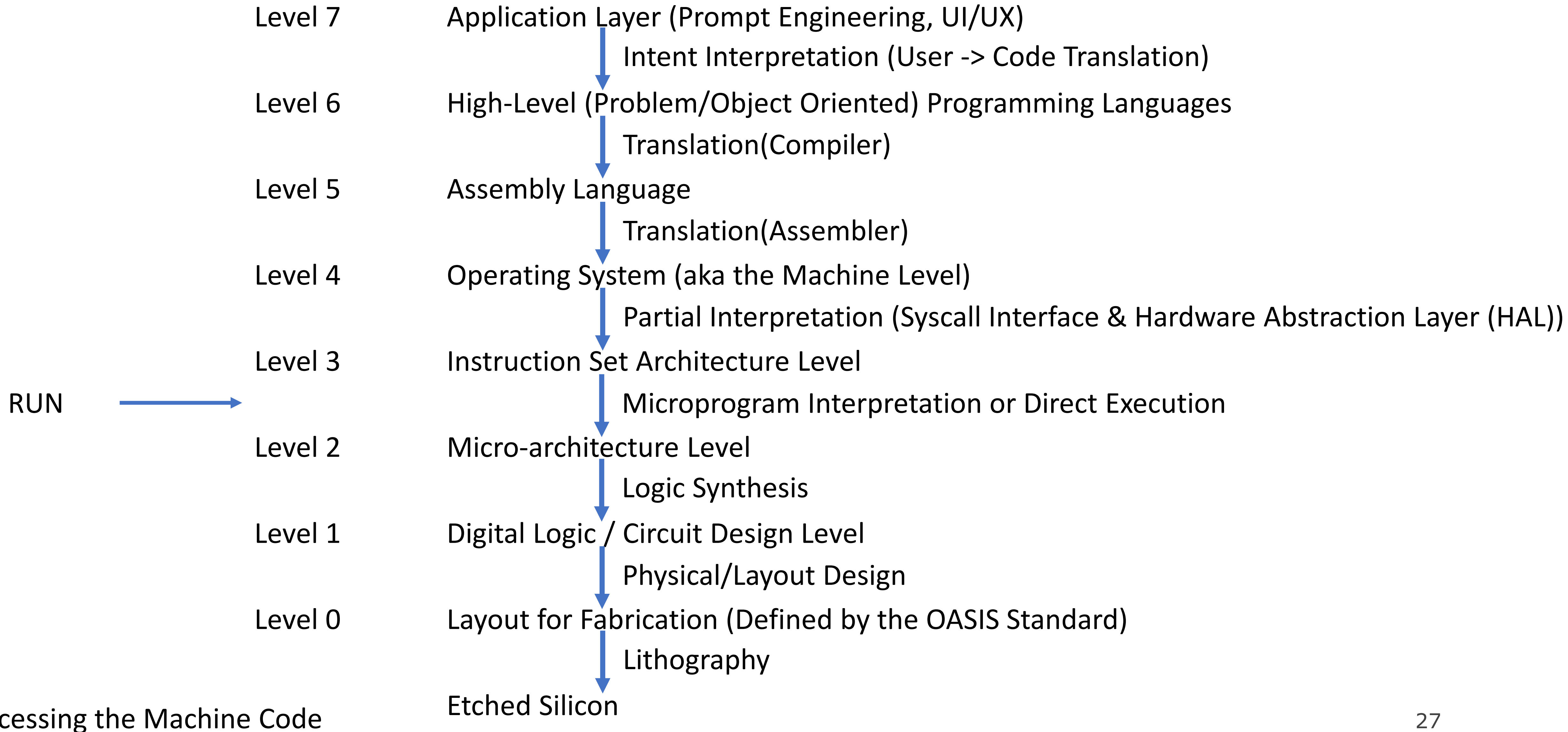


GNU Assembler (Inside GCC)

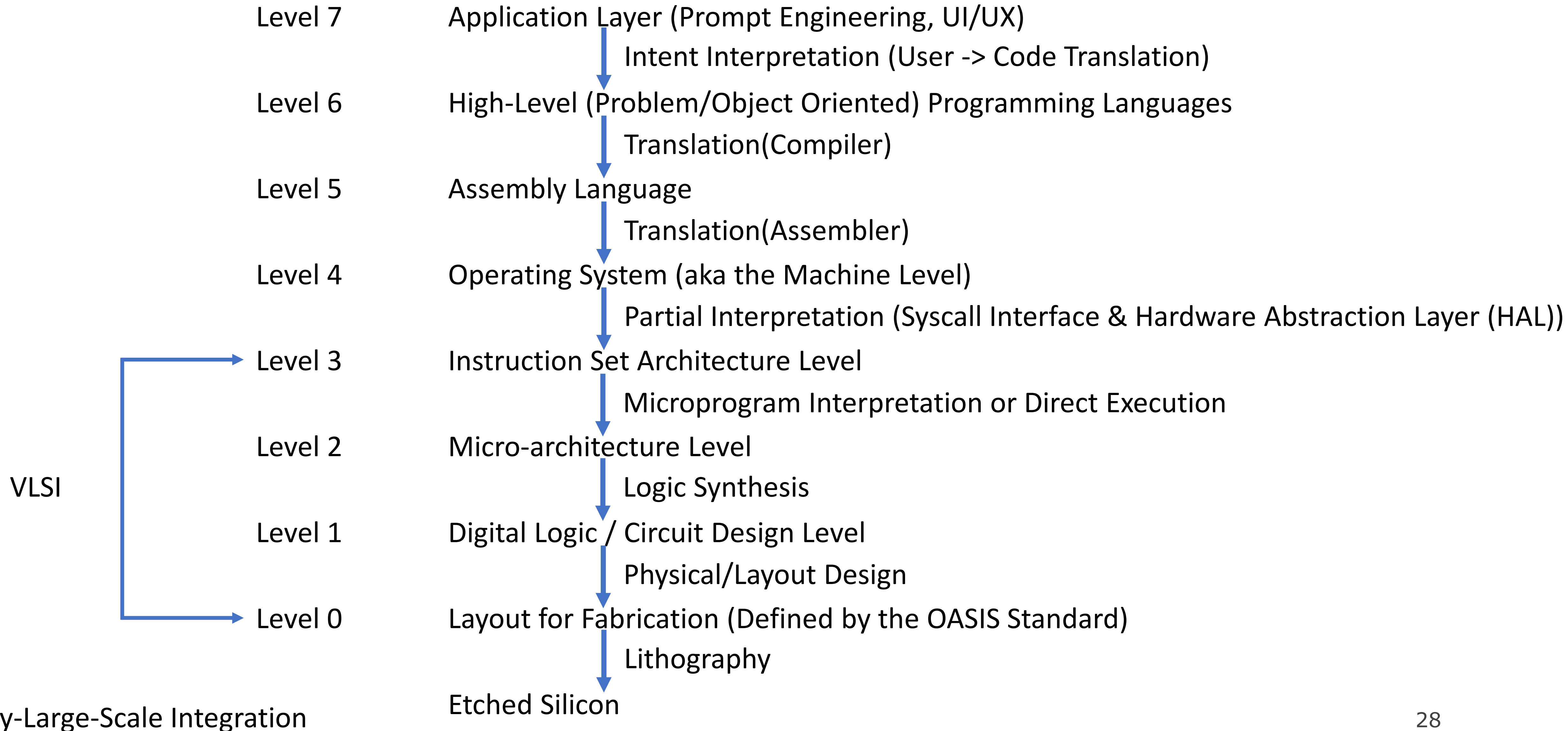
Programming Levels



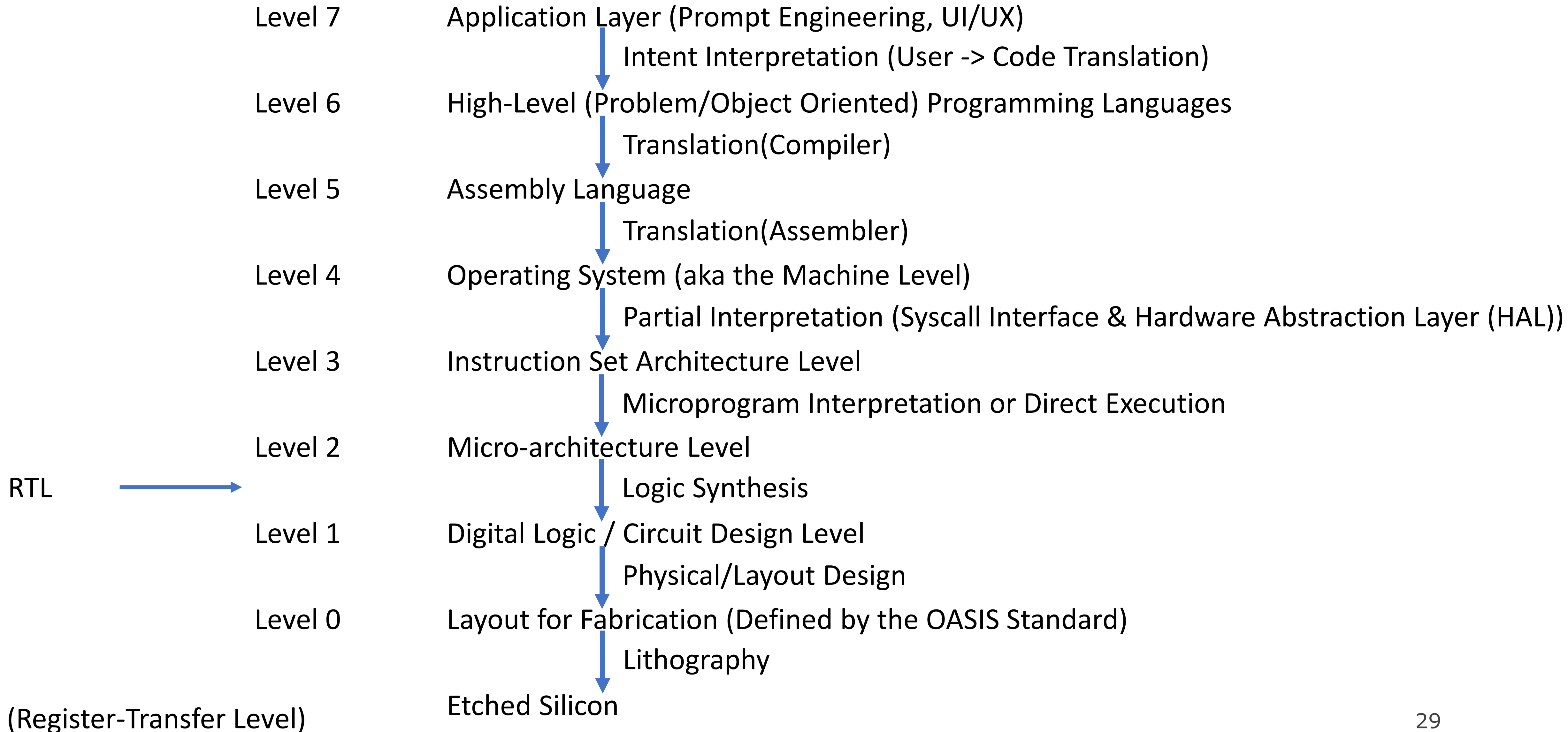
Programming Levels



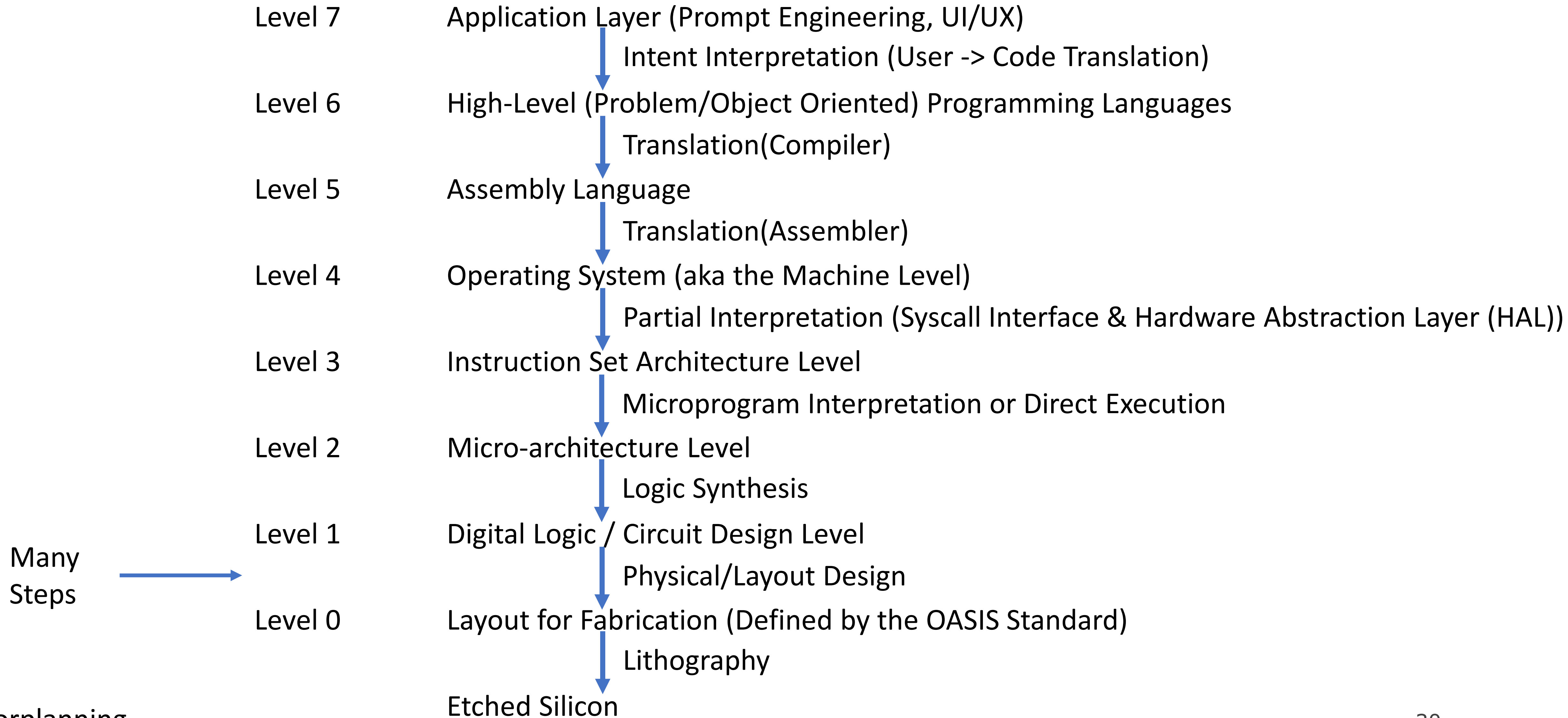
Programming Levels



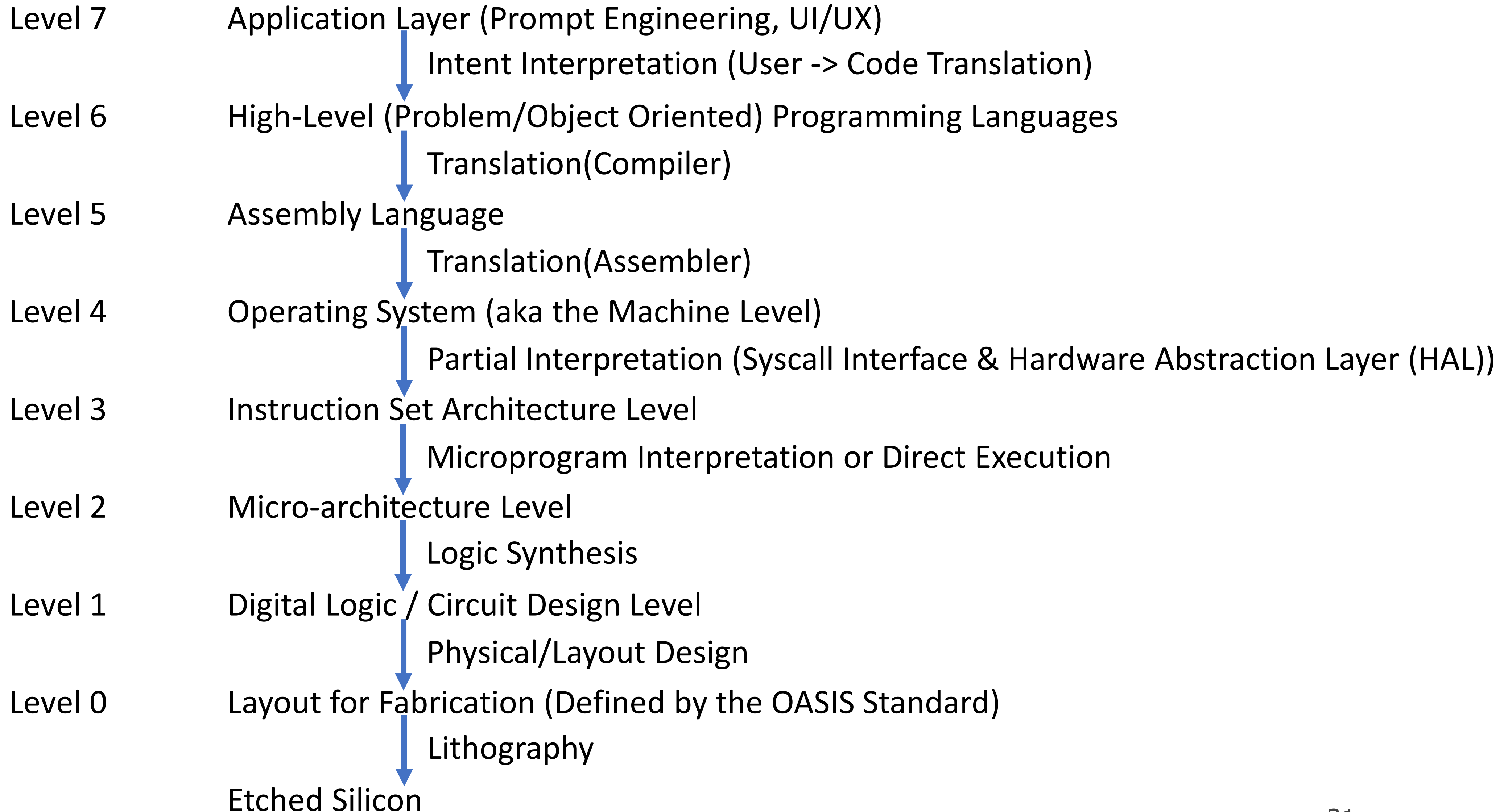
Programming Levels



Programming Levels



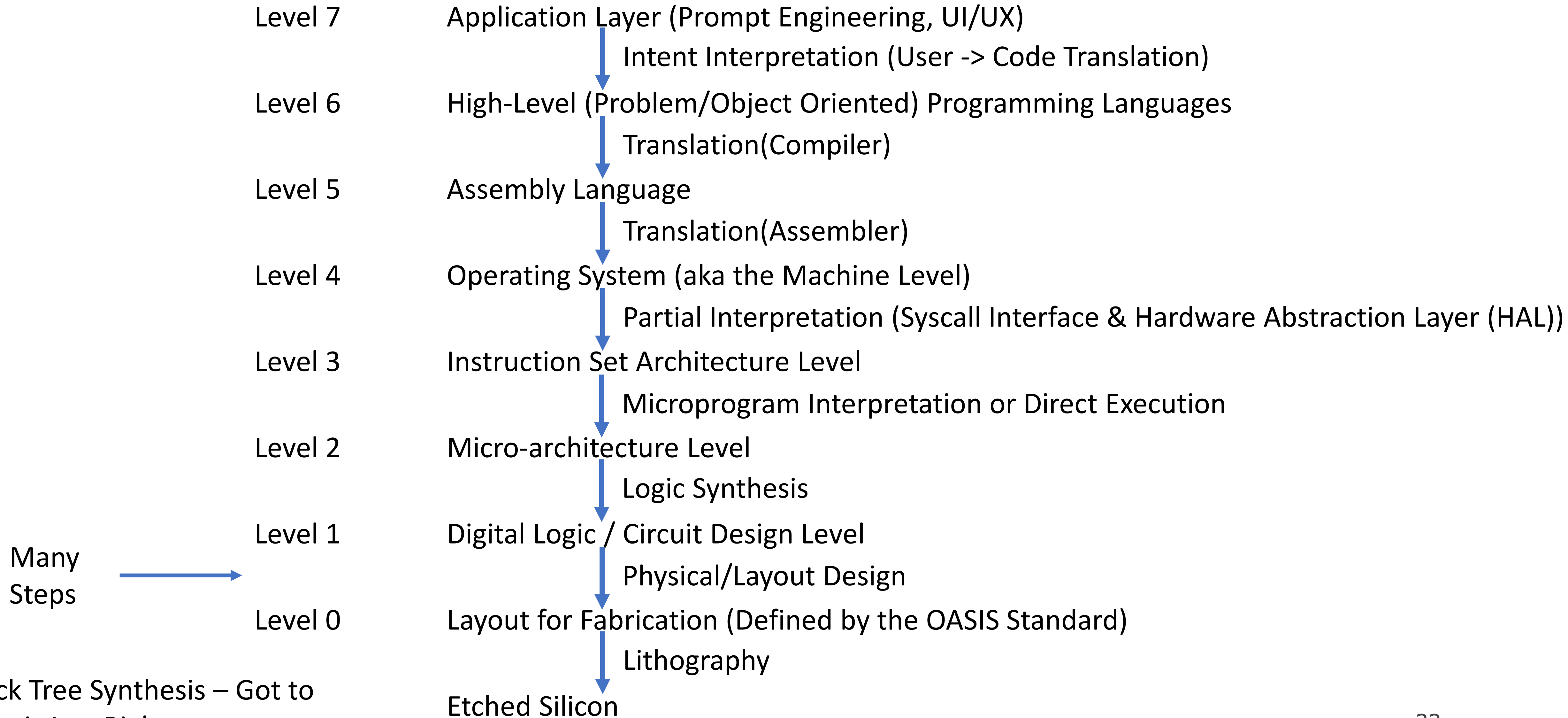
Programming Levels



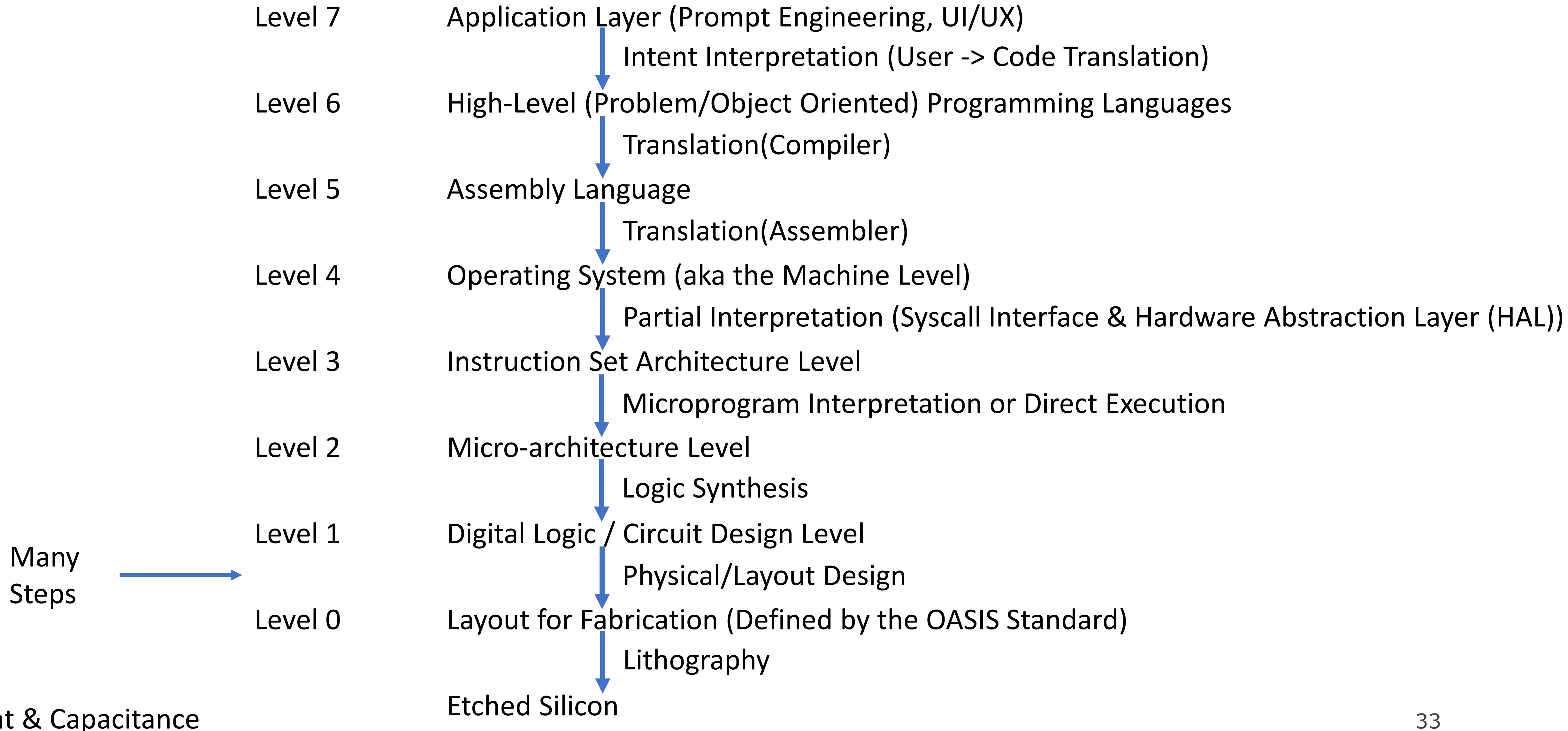
Many Steps

Wire Routing
– Don’t Cross the Wires

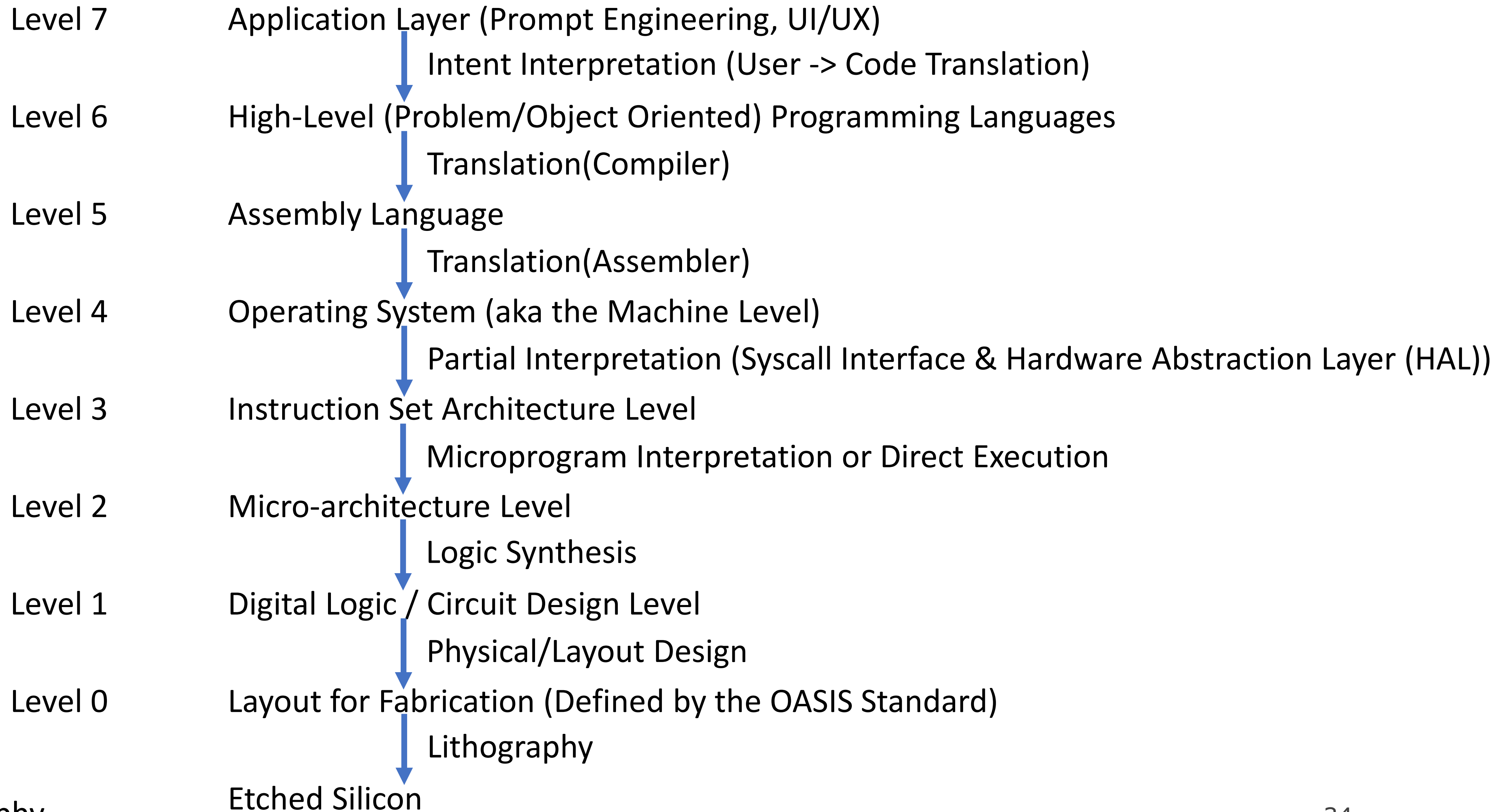
Programming Levels



Programming Levels

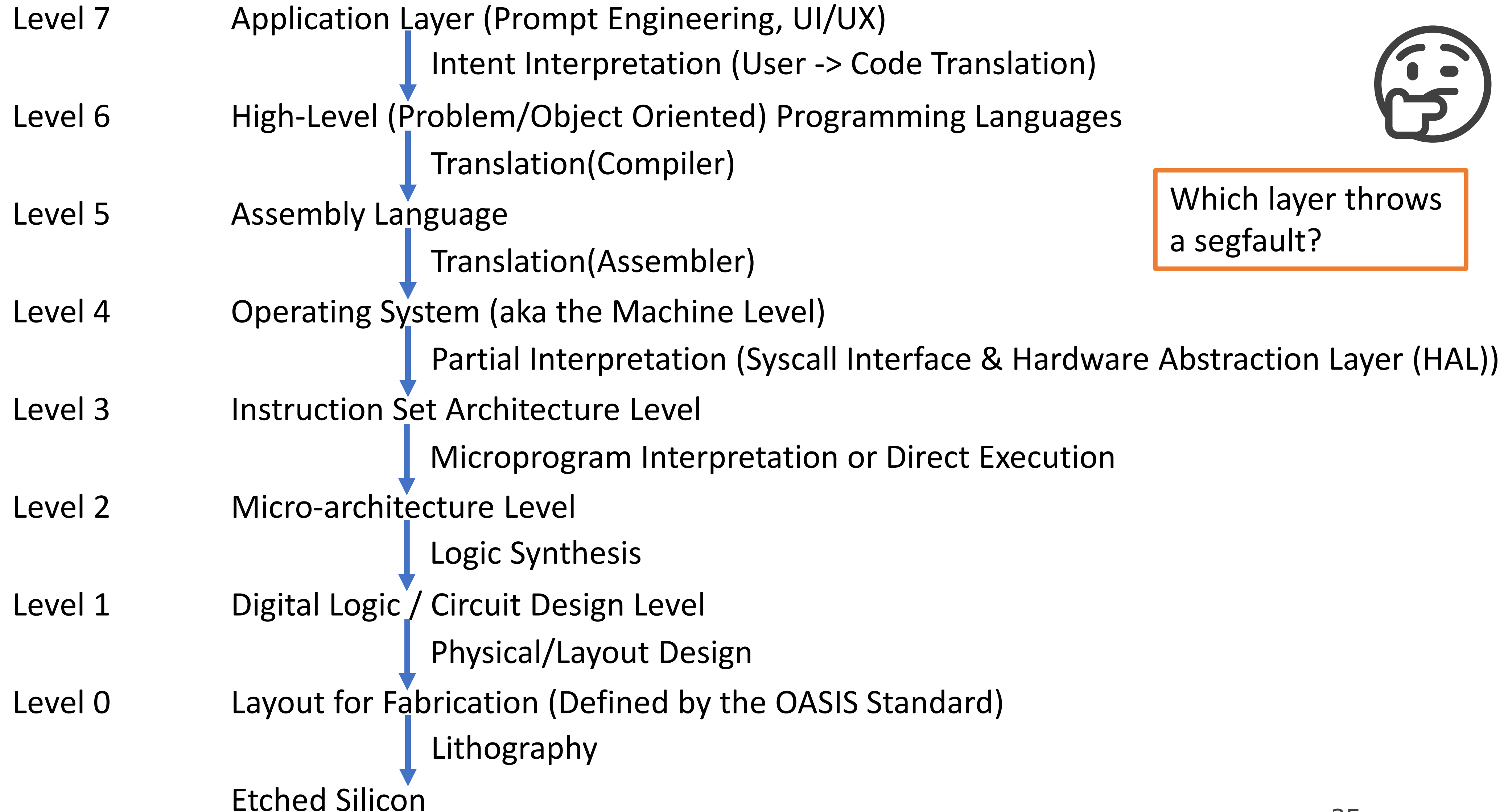


Programming Levels

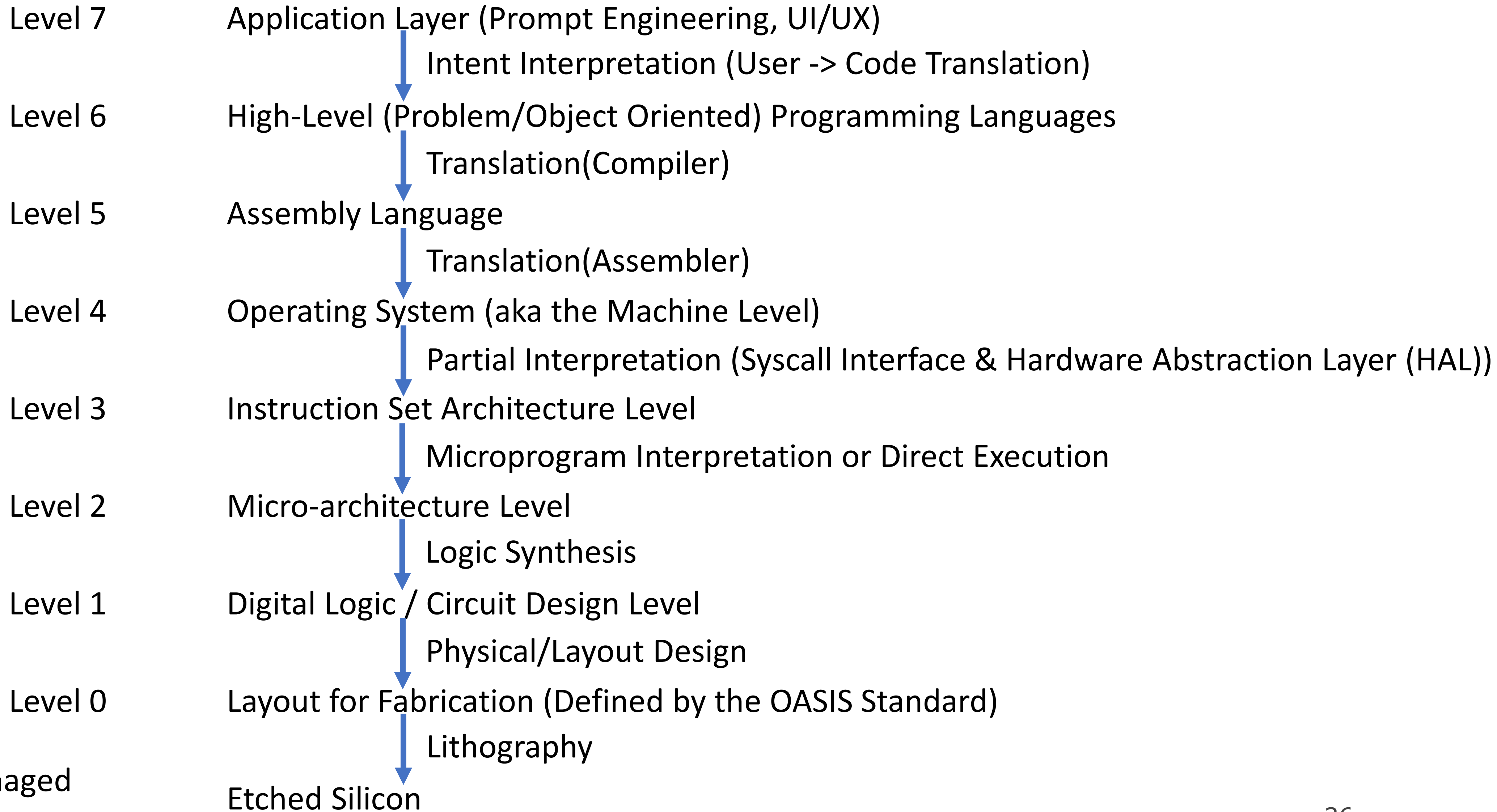


Checkout EUV Lithography

Programming Levels



Programming Levels



HAL IS WATCHING

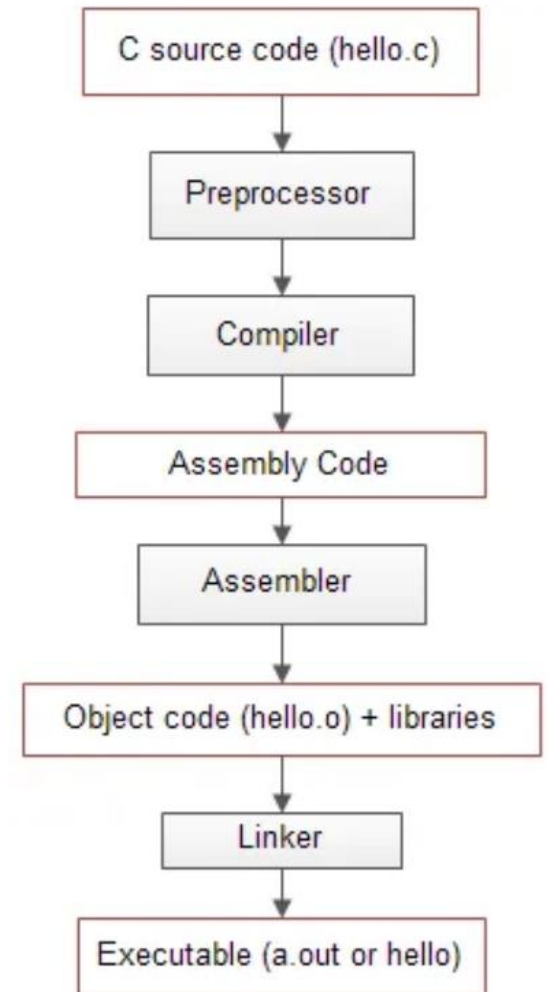
Program Memory Managed By The OS

More on the Compiler

How Does GCC Work?

- One Unix Command – A lot of steps!

gcc hello.c -o hello

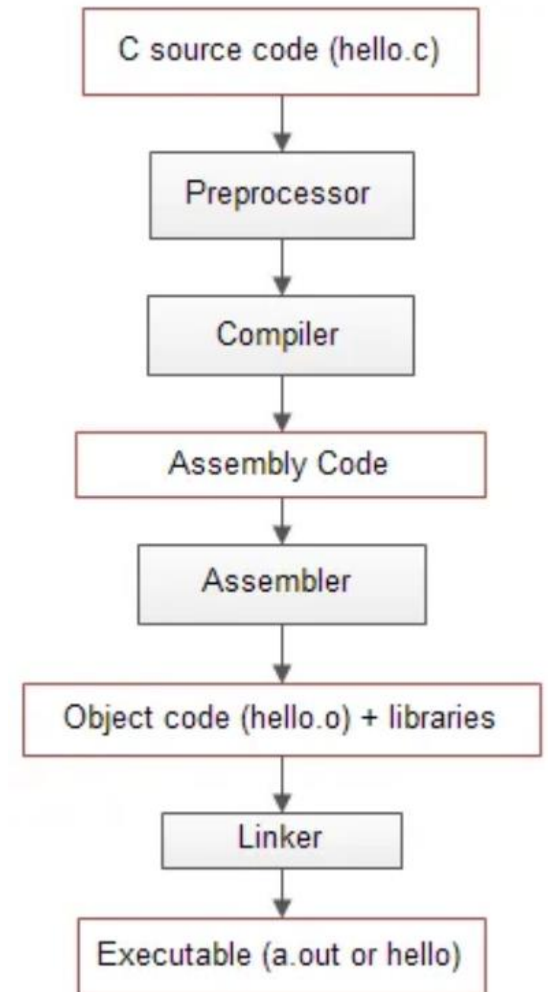


https://medium.com/@tuvo1106/the-gcc-compilation-process-8accb463e227

How Does GCC Work?

- Preprocessing – Handle Programmer Conveniences
  - #Macros convert to normal C code
  - Lines split by \ are joined
  - Comments are removed
    - NOTE: Some comments are added, but our comments are removed
  - Bring in functions and variables from the headers
    - This is how the #include is resolved

```
gcc -E hello.c > pre_processed_hello
```



https://medium.com/@tuvo1106/the-gcc-compilation-process-8accb463e227

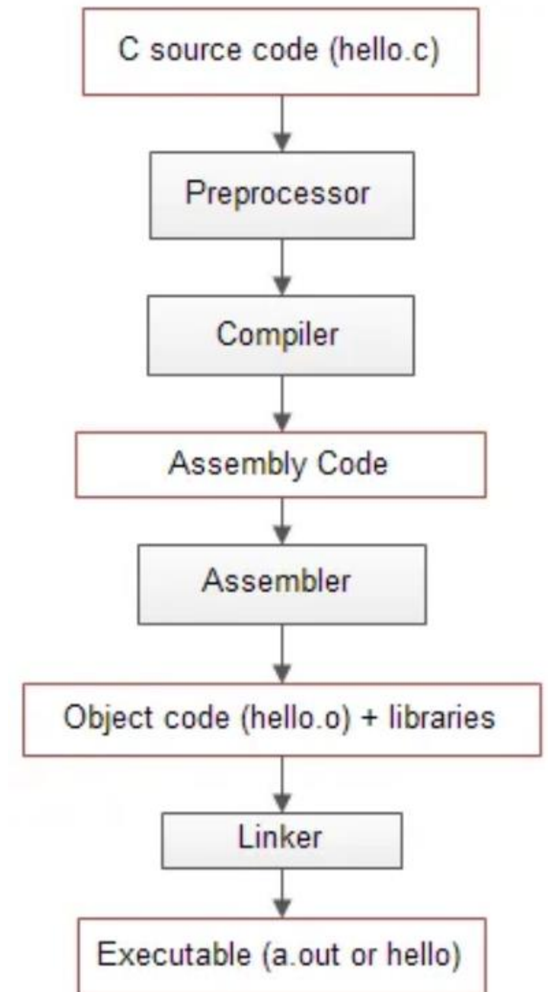
How Does GCC Work?

- Compilation – C to Assembly

gcc -S hello.c

- Will generate intermediate ‘human-readable’ assembly

- There are different styles/syntax for x86, we use AT&T
  - AT&T is also the gcc default



https://medium.com/@tuvo1106/the-gcc-compilation-process-8accb463e227

How Does GCC Work?

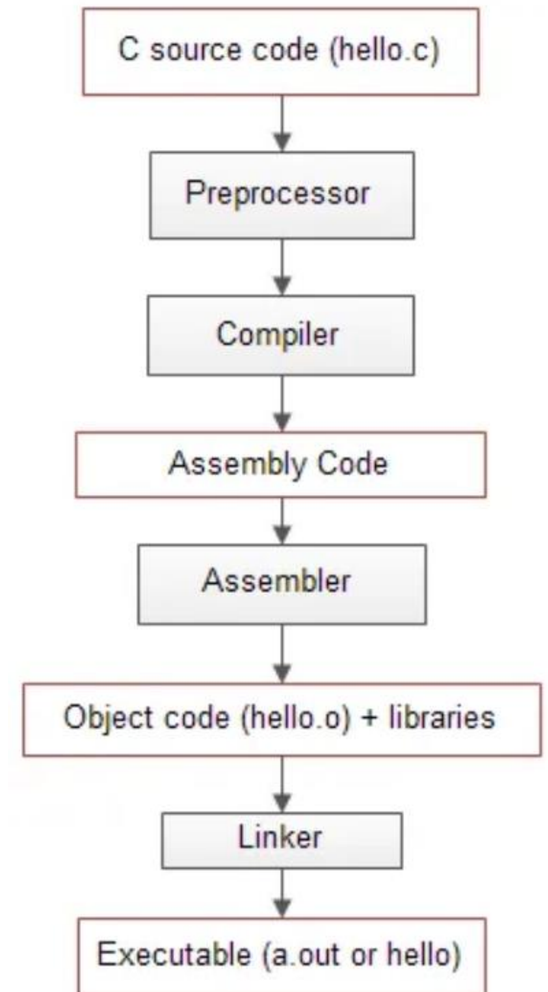
- Object Generation – C to Object File

gcc -c hello.c

- “Just compile; Don't link"

- This outputs a non-human readable Object File
  - It is defined as a type of incomplete machine code
  - With extra metadata to power linking

- Using objdump –d hello.o , we can see the assembly



https://medium.com/@tuvo1106/the-gcc-compilation-process-8accb463e227

How Does GCC Work?

- Linking – Bringing All the pieces together
  - Object Files & Libraries -> Fully Executable Machine Code

gcc hello.o -o hello

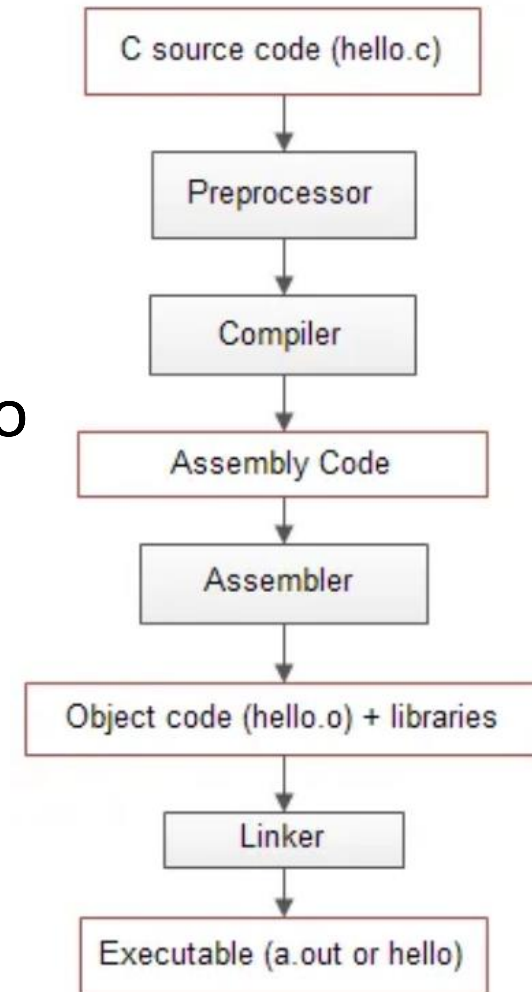
ld -o hello hello.o -lc -dynamic-linker /lib64/ld-linux-x86-64.so.2 /usr/lib/x86_64-linux-gnu/crt1.o /usr/lib/x86_64-linux-gnu/crti.o /usr/lib/x86_64-linux-gnu/crtn.o

- NOTE: We can get our .o in more than one-way

gcc -c hello.c

OR

as hello.s



https://medium.com/@tuvo1106/the-gcc-compilation-process-8accb463e227

What does the Assembler Do?

A Two Step Process

- Pass 1: Setup Memory Addresses
  - The program reads in the assembly program identifying and tracking:
    - Labels
    - Literals
    - Data Variables

- Pass 2: Generate the Machine Code (Byte/Binary Code)
  - Identify Opcode from the mnemonic assembly
  - Resolve labels/literals/variables using the tables from Step 1
  - Convert Data to Binary
  - Identifies External (Out of Program) References and places markers for the Linker
  - Setup Metadata for linking if this program has loadable parts

Final Output is not runnable, but has all the parts need if linking can complete

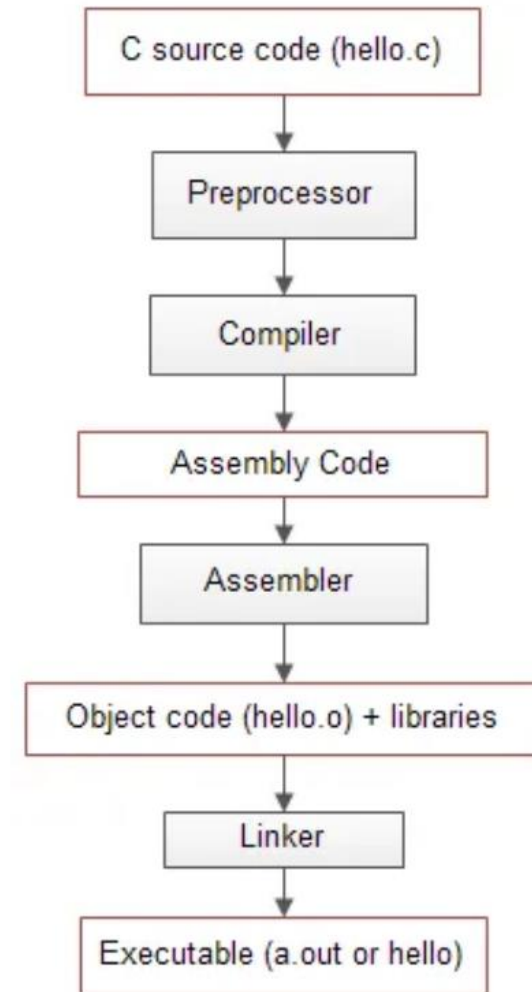
Why do we need a linker?

Many Links

- Every C file corresponds to a .o
- Libraries can also be made into linkable formats
- We don’t want to have to write all our code in 1 file and we want to use the STL
- The linker makes this all possible

How Does GCC Work?

- Multi-Step Process -> Multiple Failure Points
- Compilation can fail for many reasons at different points
- Mainly two areas that fail ‘Compilation’ or Linking
- If compilation succeeds, Intermediate Assembly will be good!

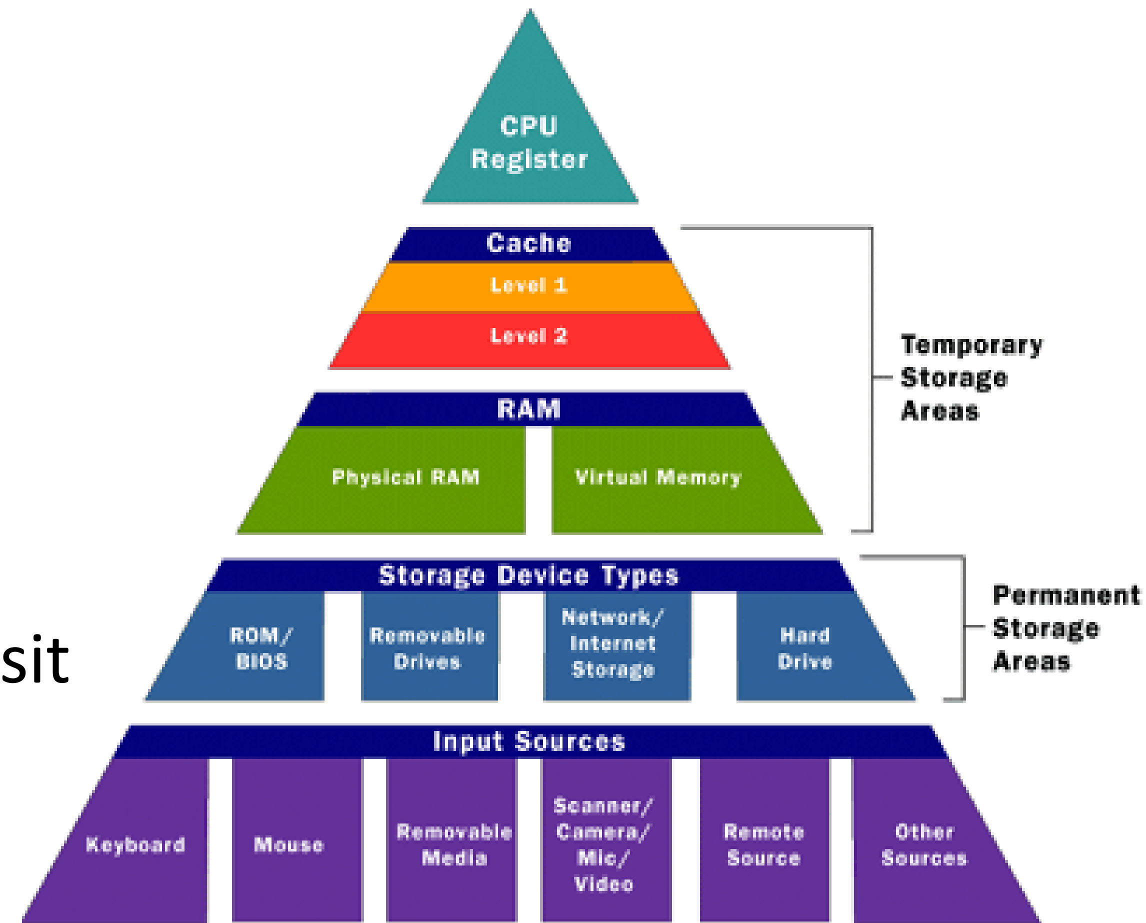


https://medium.com/@tuvo1106/the-gcc-compilation-process-8accb463e227

Peeking at Memory

Speed vs Space

- CPU is the most important place
  - Closer to CPU, less travel time
  - But limited space, so bottleneck getting there
- Think of the CPU like downtown, generally expensive and highly desirable real estate
- The BUS (actual technical name) is our transit system around the computer
- Places close to the CPU are more limited and more valuable, since they can get to the CPU faster

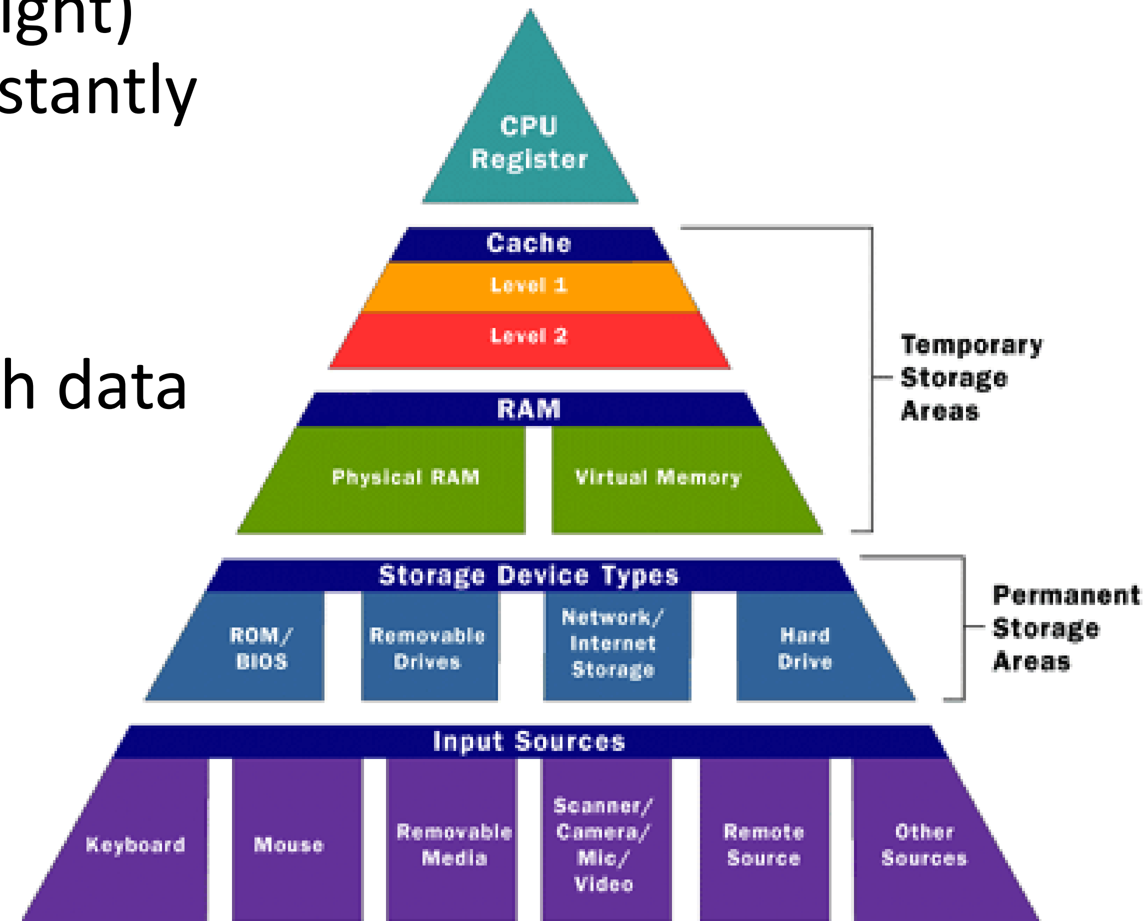


Speed vs Space

- All of Memory (Temporary Storage on the right) and the registers is rent only, so data is constantly moving around

- Many algorithms developed to decide which data gets to live where and for how long

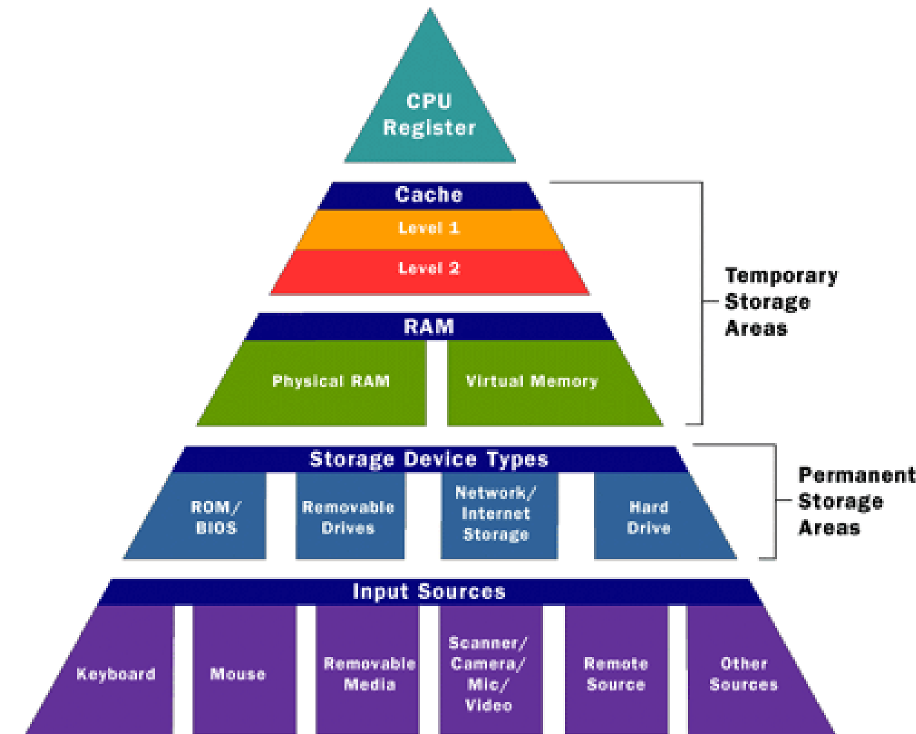
- Proper access makes a huge difference on performance



Speed vs Space

- Approximate Access Times

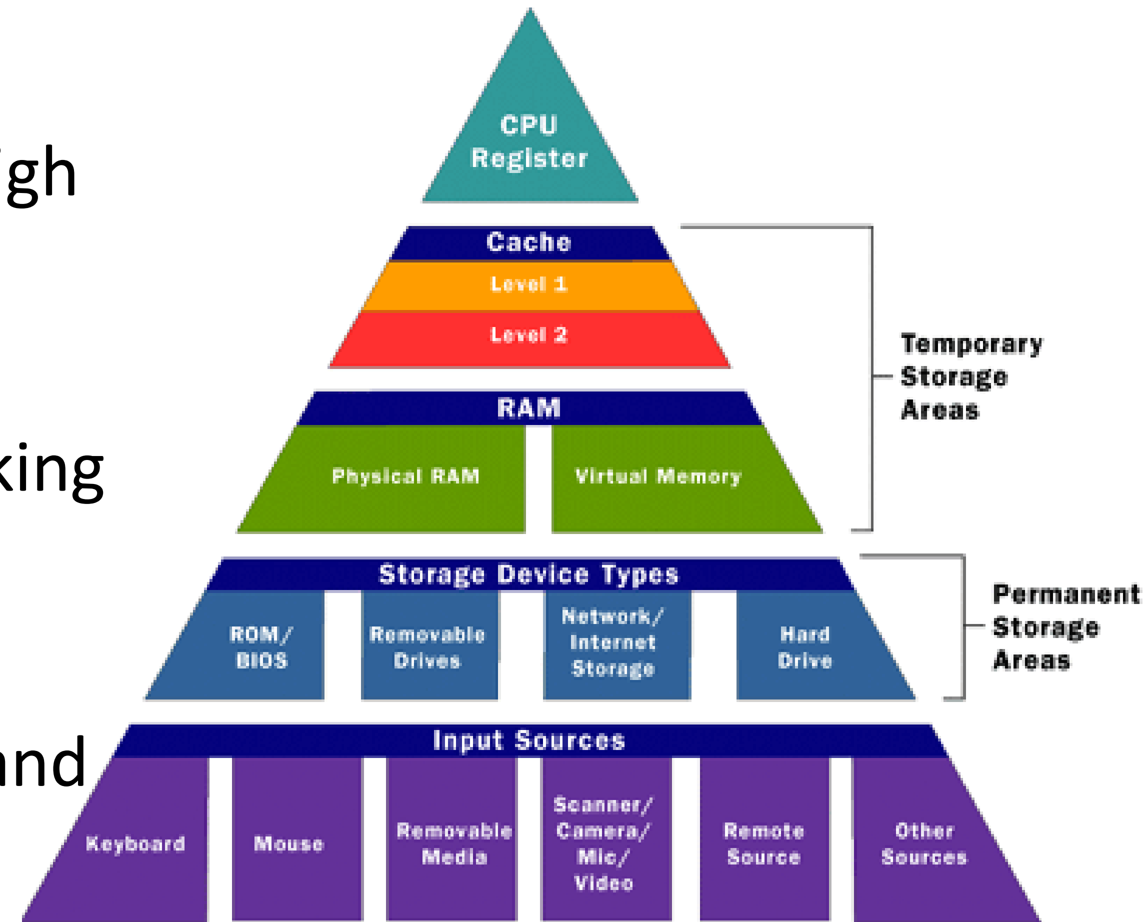
Resource	Latency Time
Register	0 Cycles (already here)
Level 1 Cache	~0.5 ns
Level 2 Cache	~7 ns (14x L1)
RAM	~100 ns (20x L2, 200x L1)
SSD	~100-150 us (~14Kx L2, 200Kx L1)
Hard (Spinning) Disk	~10 ms (~2.8Mx L2, 40Mx L1)
Network Packet CA -> Netherlands -> CA	~150 ms (~21Mx L2, 300Mx L1)
Average Human Response Time to Visual Stimulus	~200 ms (~28Mx L2, 400Mx L1)



For more on speed checkout:
https://www.cs.princeton.edu/courses/archive/spring20/cos217/lectures/20_Mem_Storage_Hierarchy.pdf
https://gist.github.com/jboner/2841832

Speed vs Space

- Pre-emptive requests and moving of data is critical
- Orders of Magnitude Improvements from high locality
- Every part of the pyramid is working on making this faster
- Better BUS, faster storage(both temporary and permanent), bigger RAM, better algorithms

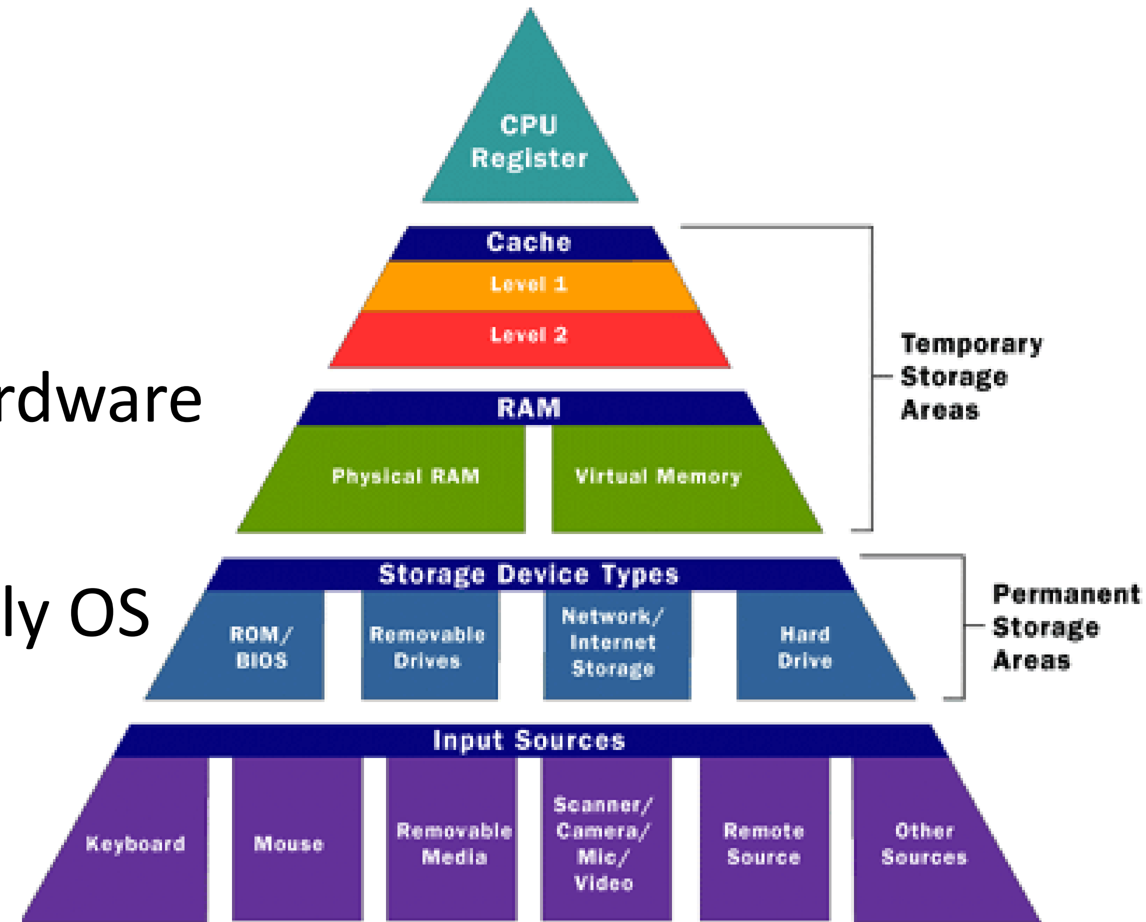


What is Locality?

- Temporal Locality
  - Has the data been used recently? Then we expect to be used again soon
- Spatial Locality
  - The data appears close together in the program/memory, so it will likely be needed at the same time.
- Hardware and OS designers consider algorithms to predict and leverage locality to optimize management of memory resources
- Cache in particular is a limited resource and must be used effectively to leverage benefits

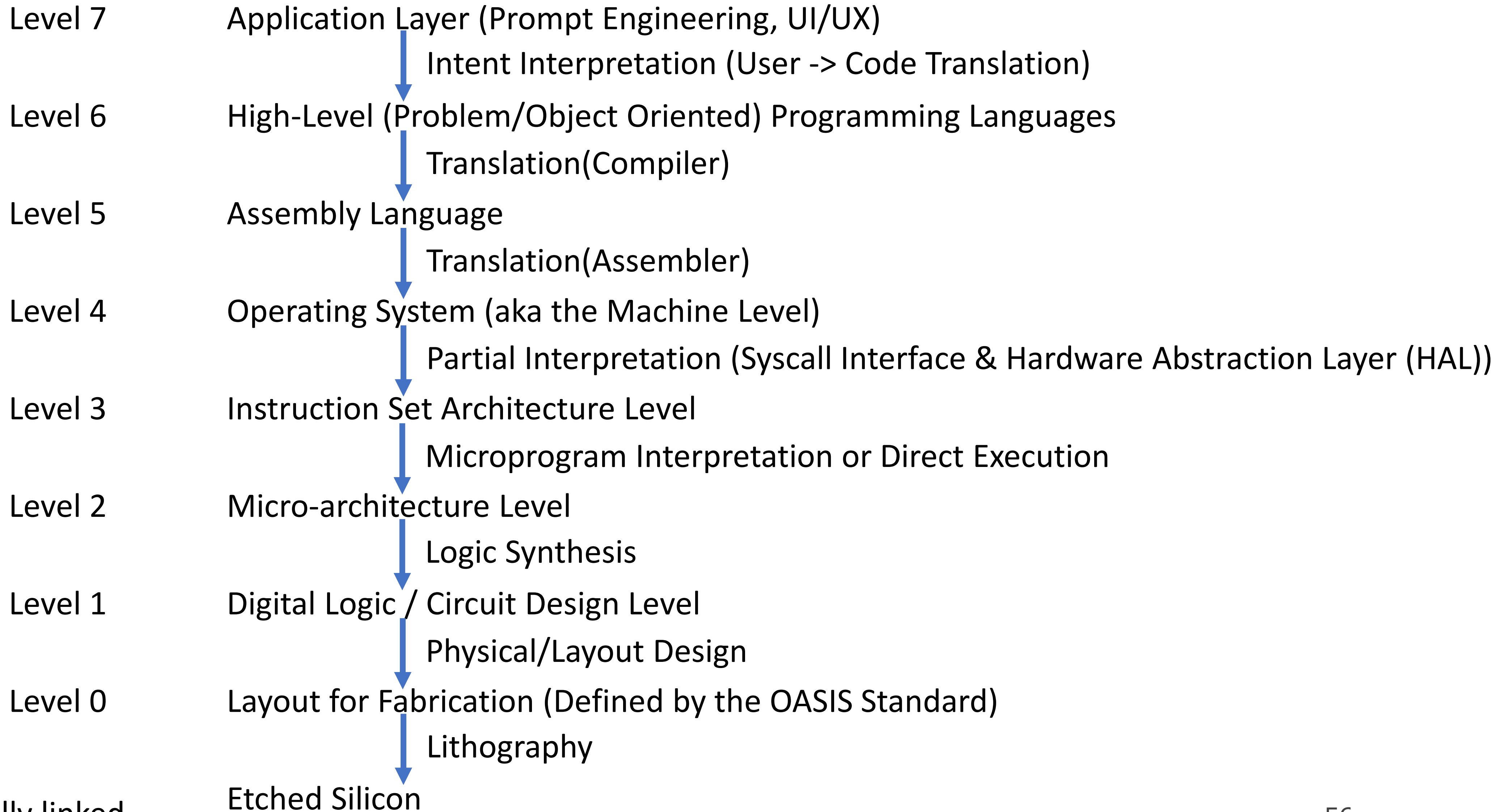
Who Gets to Manage the Memory?

- Registers – Managed by the Compiler/Assembler
- Cache – Managed by Hardware Designers
- Memory – Mainly the OS, influenced by hardware
- Disk – Managed by the user and occasionally OS



Architecture & The ISA

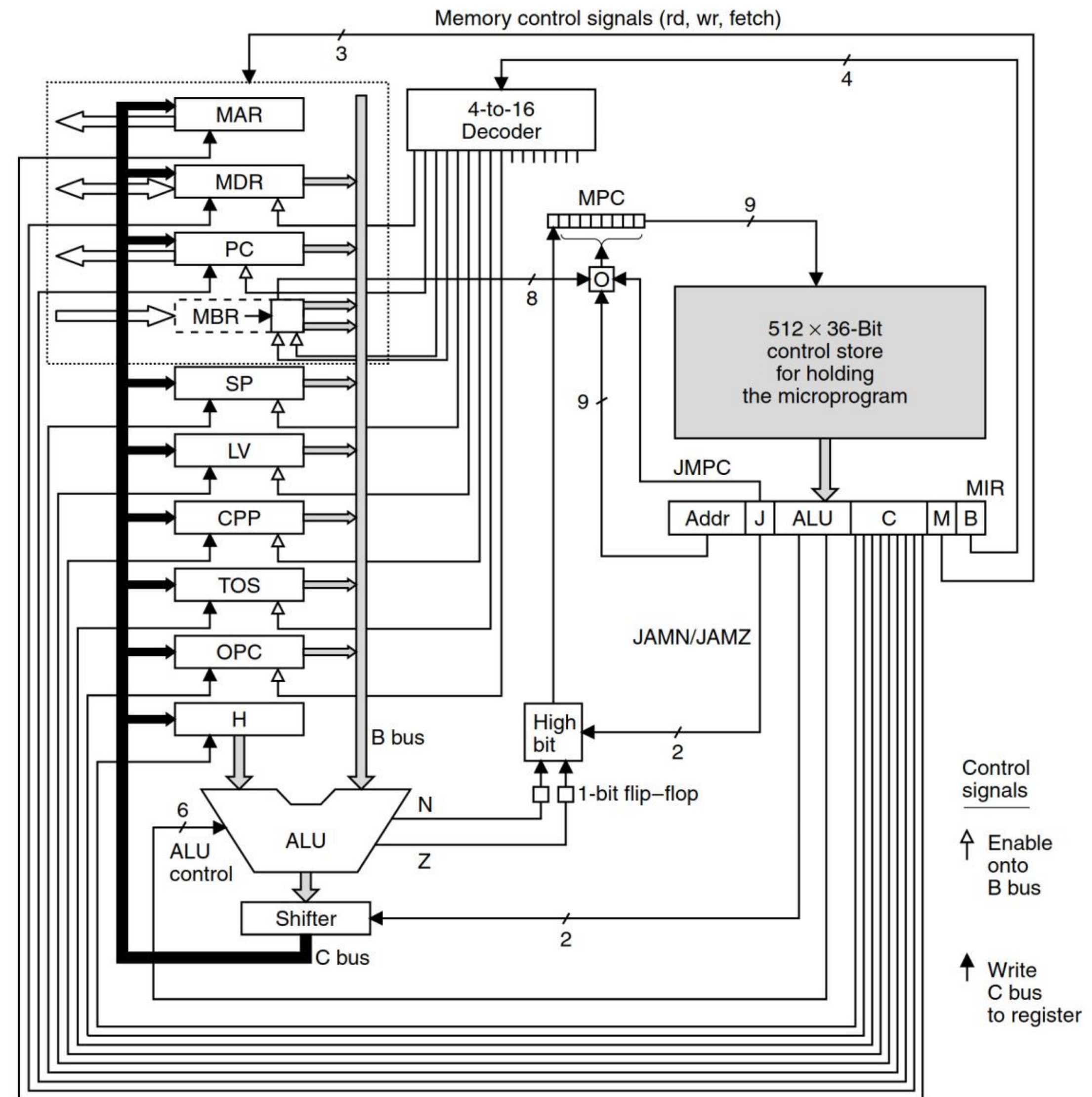
Programming Levels



These levels are integrally linked

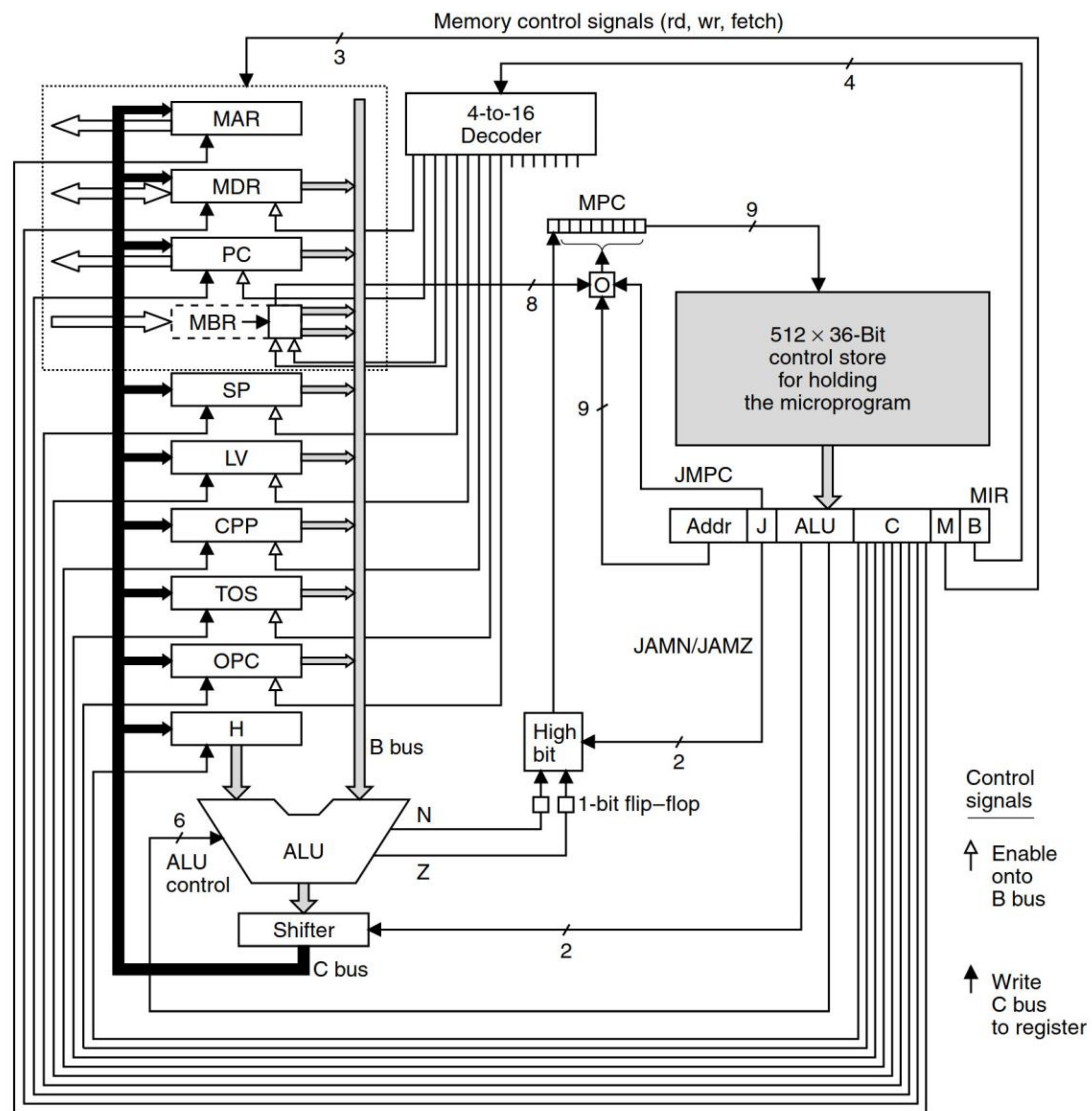
A 'Simple' Example

- MIC-1 Architecture (Tanenbaum - Structured Computer Organization 6th Edition)
- IJVM ISA – Subset of the Java Virtual Machine
- A 'Vanilla' processor design



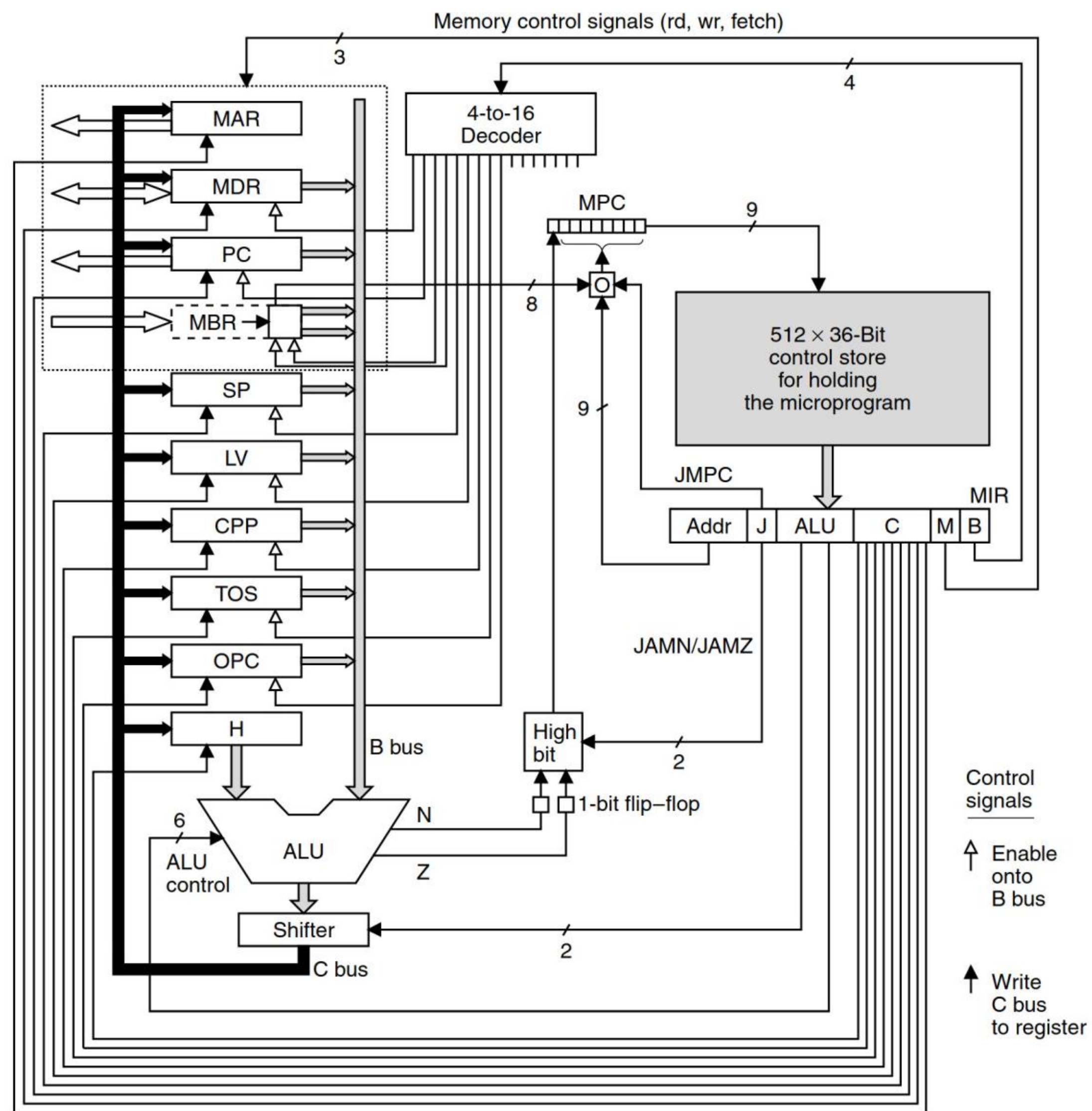
A ‘Simple’ Example

- Control Store is the most important part!
- Our ISA is defined by that unit
- 9 wires in -> 2**9 possible combinations, 2**9 (512) possible commands
- Each command drives 36 wires to control the chip
- Assembly/Machine Language is defined by the hardware



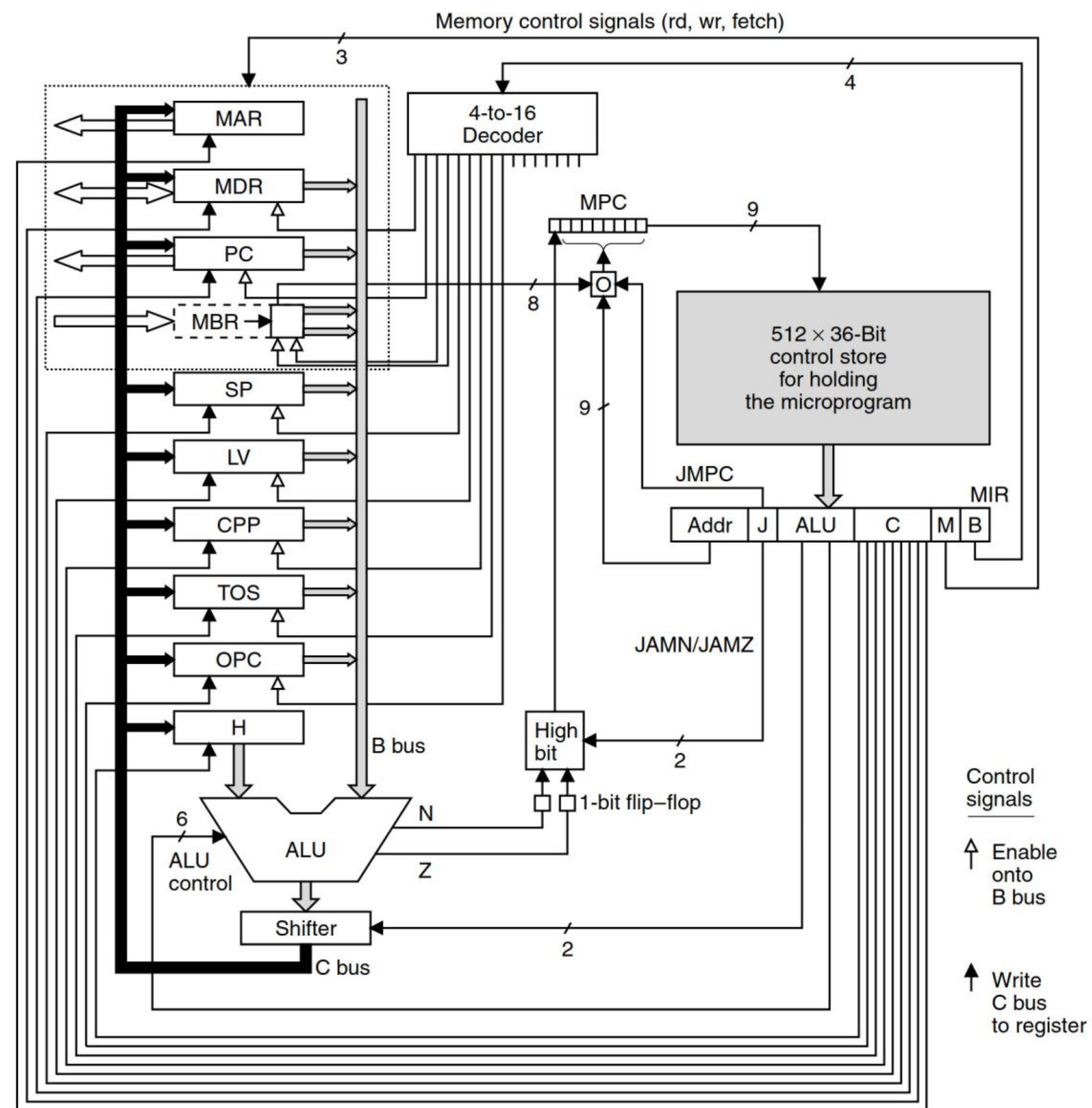
A 'Simple' Example

- ALU – Arithmetic & Logic Unit
  - Performs Math & Logic Operations
- MAR – H are the registers
- B + Decoder – Enables Register to load onto B Bus
- Z and N act similar to our condition codes, but in a much more limited/simple way
- C controls the C Bus, informing the destination register to receive its value



A ‘Simple’ Example

- Notice how the ALU is only able to take in the left operand from the H register
- All two operand ALU operations, would need to first load the left operand to H
- This would be an example of a hardware based constraint



Better Design Better Performance

- The MIC-2 Fixes this issue by adding another BUS improving the Datapath

- Design directly impacts the ISA that we can make available