1. Computer Architecture Patent – What is it?

A computer architecture patent protects innovations related to the design and structure of computer systems. This can include:

- Novel processor designs (e.g., pipeline architecture, cache organization)
- Instruction sets or data paths
- Power-efficient computing methods
- Hardware accelerators (e.g., for AI or graphics)
- Memory management systems

To be patentable, the architecture must be:

- Novel (not previously known)
- Non-obvious to someone skilled in the field
- Useful

2. How to Get a Patent (General Process):

Step-by-Step:

1. Document your invention – include diagrams, architecture, algorithms, etc.
2. Search for prior art – use databases like Google Patents or the USPTO to see if your idea already exists.
3. Determine the type of patent:
    - Utility (most common for computer architecture)
    - Design (for product appearance)
    - Provisional (temporary placeholder, gives you 12 months)
4. Draft a patent application – includes claims, descriptions, drawings.
5. File with a patent office:

    - USPTO (United States)
    - CIPO (Canada)
    - EPO (Europe)
6. Undergo examination – your patent may be accepted, rejected, or need modification.
7. Pay fees – for application, maintenance, etc.

Important: It's best to work with a patent attorney.

3. What is the Physics Behind a Computer?

A computer is a physical machine operating under the laws of physics—mainly:

- Electromagnetism: governs how signals move through wires and how transistors switch.
- Semiconductor Physics: how electrons behave in materials like silicon (basis for transistors and integrated circuits).
- Thermodynamics: deals with heat generation and dissipation.
- Quantum Mechanics: at very small scales, especially in modern nano-scale transistors.
- Classical Mechanics: for physical parts (e.g., fans, hard drives with spinning disks).

At its core, a computer manipulates electrical voltages that represent binary data (0 and 1), using logic gates built from transistors.

4. What is the Physics Behind a Vacuum Tube?

Vacuum tubes (used before transistors) operate on thermionic emission:

- Inside the tube is a vacuum.
- A cathode is heated (usually with a filament), causing it to emit electrons (due to thermionic emission).
- These electrons travel across the vacuum to the anode (positively charged plate).
- A grid in between controls the flow of electrons—this allows amplification or switching.

Key Physics Concepts:

- Thermionic Emission (electron release due to heat)
- Electric Fields in a vacuum
- Control Grids modulating electron flow (acts like a gate)
- Vacuum Physics (no air means fewer collisions for electrons)

1. Computer Architecture Patent

A computer architecture patent protects new designs or methods in how computers are built or operate. This includes things like processor design, memory structure, instruction sets, and hardware accelerators. To be patentable, it must be new, non-obvious, and useful.

2. How to Get a Patent

1. Document the invention — include technical details, diagrams, and how it works.
2. Search for existing patents — make sure your idea hasn't already been done.
3. Choose patent type — most computer-related ideas use utility patents.
4. Write the patent — include claims and a detailed description.
5. File it — submit to the patent office (e.g., USPTO, CIPO).
6. Examination — the office reviews and may ask for changes.
7. Pay fees — both upfront and ongoing maintenance.

Best to work with a patent lawyer.

3. Physics Behind Computers

Computers rely on physics to work. Key areas:

Electromagnetism — moves signals through circuits.

It governs how electric fields and currents behave, allowing information to be transmitted and processed.

Semiconductor physics — explains how transistors switch on/off.

It's based on controlling the flow of electrons in materials like silicon using electric fields.

Quantum mechanics — matters in small-scale chips.

At the nanoscale, electrons don't behave classically, so quantum effects like tunneling impact how modern transistors work.

Thermodynamics — deals with heat produced by components.

As components process data, they generate heat that must be managed to prevent damage or slowdown.

At the hardware level, computers use voltage changes to represent binary 0s and 1s, processed through logic gates made of transistors.

4. Physics Behind Vacuum Tubes

Vacuum tubes work by heating a cathode to release electrons (thermionic emission). These electrons move across a vacuum to an anode. A control grid in between can stop or allow the flow, making it act like a switch or amplifier.

Key concepts:

- Thermionic emission — heat releases electrons.
- Electric fields — push electrons through the vacuum.
- Vacuum — no air, so electrons move freely.
- Control grid — regulates current flow.

Binary Addition

Concept Overview:

- Binary addition mirrors decimal addition, but digits are restricted to 0 and 1.
- The four possible bit-level additions:
    - 0 + 0 = 0

- 0 + 1 = 1
- 1 + 0 = 1
- 1 + 1 = 10 (binary for 2 → results in 0 with a carry of 1)

Carry Propagation:

- Carry bits must be added to the next leftward column, similar to decimal carryover.
- E.g., 1 + 1 + 1 = 11 (binary for 3 → sum is 1 with a carry of 1)

Overflow:

- Occurs when a result exceeds the fixed number of bits allowed (e.g., 8 bits).
- The carry is lost or wraps around, leading to incorrect results.

Adder Circuits

Half Adder

- **Inputs**: A and B (single bits)
- **Outputs**: Sum (S) and Carry (C)
- **Logic**:
  - S = A XOR B
  - C = A AND B
- **Limitation**: Cannot process carry-in from a previous addition step.

Full Adder

- **Inputs**: A, B, and Cin (carry-in)
- **Outputs**: Sum (S) and Cout (carry-out)
- **Logic**:

    - Sum = A XOR B XOR Cin
    - Cout = (A AND B) OR (B AND Cin) OR (A AND Cin)

Ripple Carry Adder

- Chains multiple full adders for multi-bit addition.
- The carry-out of one adder connects to the carry-in of the next.
- **Drawback**: Time delay increases with each stage (carry ripple delay).

Fast Carry Adder

- Optimizes carry computation (e.g., carry look-ahead).
- Each stage computes carry based on input bits directly, improving speed.

Signed Number Representations

Signed Bit Representation

- MSB (Most Significant Bit) indicates the sign:
    - 0 → non-negative
    - 1 → negative
- Simple conceptually, but breaks standard arithmetic rules.

1’s Complement

- Positive: Same as unsigned.
- Negative: Flip all bits of the positive number.
- **Issues**:
    - Two representations of zero (+0 and -0).
    - Carry bits may need to be added back to LSB → extra logic.

2’s Complement

- Most common representation in computer systems.
- To negate a number:
    - Invert all bits (1’s complement).
    - Add 1.
- **Advantages**:
    - Single representation of zero.
    - Arithmetic operations are consistent and hardware-friendly.

Binary Subtraction

- Performed using addition of 2’s complement:
    - A - B = A + (-B)
- **Negating B** involves:
    - Inverting all bits.
    - Adding 1.
- **Circuit-Level Implementation**:
    - Use XOR gates with 1 to invert bits.
    - Add 1 using a full adder circuit.

Overflow

Unsigned Overflow

- Occurs when sum exceeds the maximum representable value (e.g., 255 for 8-bit).
- Easily detected using carry-out bit.

Signed Overflow

- Trickier: Carry-out doesn't always mean overflow.
- **Detection Rule**:
    - If adding two positives yields a negative → overflow.
    - If adding two negatives yields a positive → overflow.
- **Check**:
    - Overflow if signs of operands are the same but differ from the result.

Shift and Rotation

Logical Shift

- Left: Shifts bits left, inserting 0 at LSB. Used for unsigned multiplication by powers of 2.
- Right: Shifts bits right, inserting 0 at MSB. Used for unsigned division.

Arithmetic Shift

- Left: Same as logical left shift.
- Right: Preserves sign bit (duplicates MSB). Used for signed division.
- **Use Case**: Multiply/divide signed numbers by powers of 2.

Multiplication via Shift

- Multiply by 13:
    - $13x = 8x + 4x + 1x$
    - Shift left by 3, 2, and 0 respectively, then add results.

Booth's Algorithm

Goal: Efficient multiplication, particularly when one operand has many 1s.

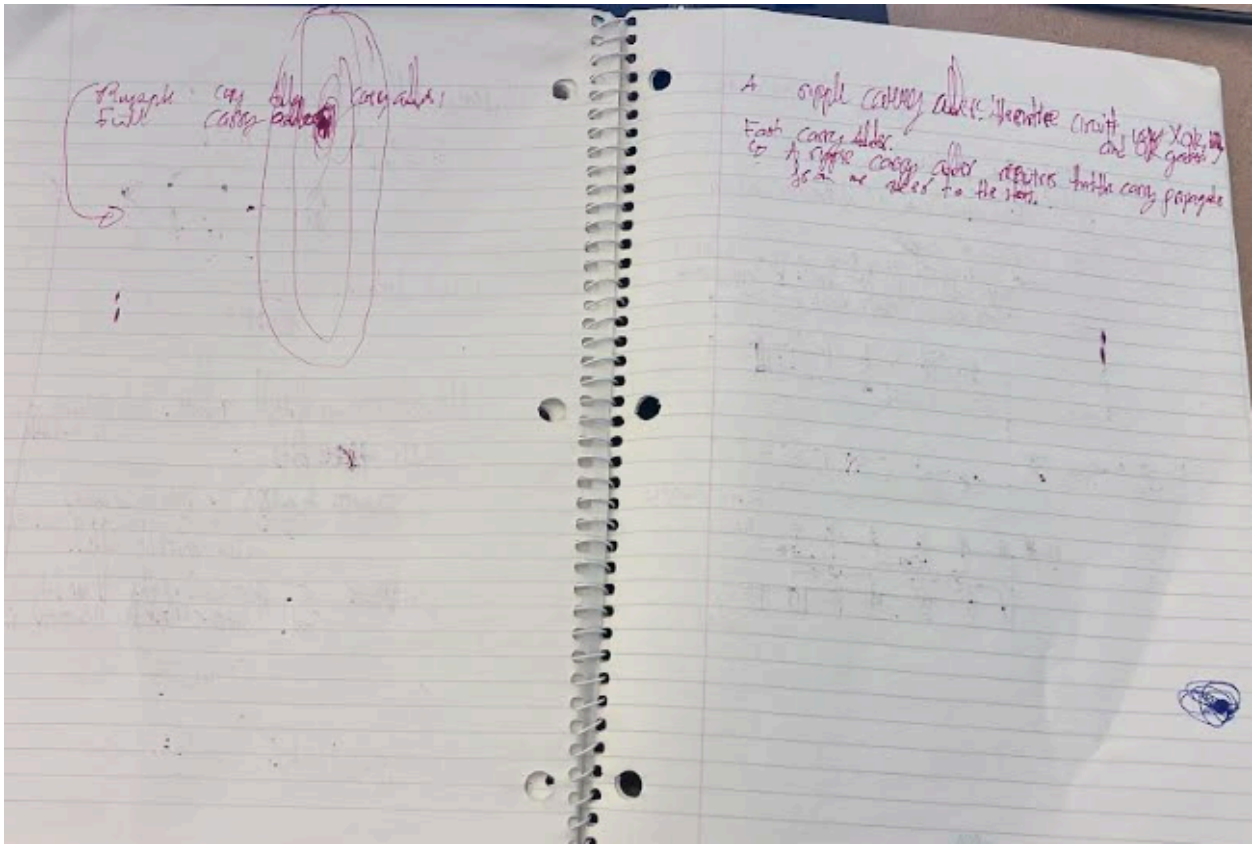
Core Components:

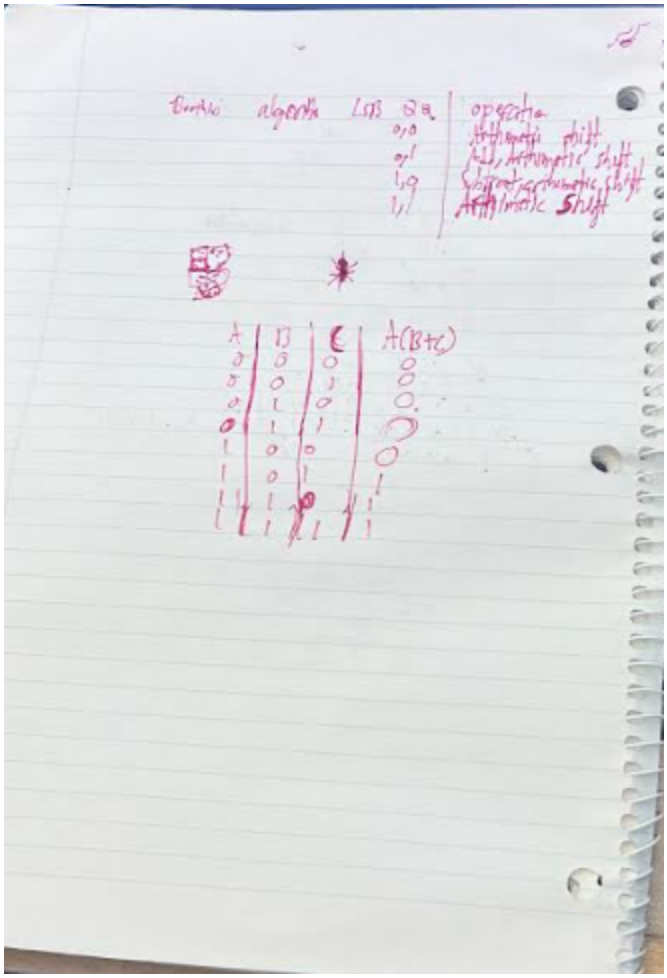
- Q: Multiplier
- M: Multiplicand
- A: Accumulator (initially 0)
- Q′: Last bit of multiplier (initially 0)

Steps:

1. Inspect Q_0 and Q′:
    - 10 → Subtract M from A
    - 01 → Add M to A
    - 00 or 11 → No arithmetic operation
2. Arithmetic right shift (A, Q, Q′ together)
3. Repeat for number of bits in Q

- Combine A and Q.
- If MSB = 1, result is negative → compute 2's complement to get magnitude.




Boolean algebra
Logic
operation
0,0
0,1
1,0
1,1
A(B+C)