

Topic 15: Hardware

Designing circuits with memory, how processors guess the future, and why GPUs are different.

Optional stretch material. This goes beyond the 9618 syllabus, so you will not be asked about it directly in the exam. Its purpose is to deepen ideas you already know. **Builds on:** 15.1 Processors and parallel processing, 15.2 Boolean algebra and logic circuits. Try the challenge questions on the last pages before you read the answers.

Finite state machines

A **finite state machine (FSM)** is a design tool for anything with memory of what has happened: vending machines, traffic lights, network protocols, even the control unit of a CPU. It has a set of **states**, **transitions** triggered by inputs, and **outputs**. In hardware, the state is held in flip-flops, and logic gates compute the next state and output.

Example: detect two 1s in a row in a stream of bits (output 1 when the latest two inputs were both 1). Two states are enough: S0 (the last bit was 0 or nothing has arrived) and S1 (the last bit was 1).

Current state	Input	Next state	Output
S0	0	S0	0
S0	1	S1	0
S1	0	S0	0
S1	1	S1	1

When the output depends on state *and* input (as here) it is a **Mealy** machine; when it depends on state alone it is a **Moore** machine.

Speculation: the CPU that guesses

Pipelining only works if the CPU keeps the stages full, but a branch means it doesn't yet know which instruction comes next. Modern CPUs use **branch prediction**: guess the outcome, keep fetching down the guessed path and **speculatively execute** it. A right guess costs nothing; a wrong guess forces the pipeline to be flushed, wasting around 15–20 cycles. Simple predictors count recent outcomes (a loop branch is taken 99 times, then not taken once, so "always taken" is right 99% of the time).

Out-of-order execution goes further: the CPU runs later instructions that don't depend on a stalled earlier one, then commits results in the original order so the program looks as if it ran in sequence.

Security cost: the **Spectre** attacks (published 2018) showed that speculative work, although discarded, can leave traces in the cache. An attacker can measure access times to infer secrets it was never allowed to read. Hardware and software fixes cost some performance: a real trade-off between speed and security.

GPUs and data parallelism

A CPU has a few powerful cores optimised for low latency on *one* complicated stream of instructions. A **GPU** has thousands of simple cores that execute the *same instruction on many data items at once* (SIMD-style). Brightening every pixel in a 4K image, multiplying large matrices (the heart of graphics and neural networks) and simulating particles are ideal. Code full of unpredictable branches, or where each step depends on the previous result, is not. (Link: Amdahl's law in the topic 4 extension limits what any parallel hardware can achieve.)

Challenge questions

1. Using the "two 1s in a row" machine above (starting in S0), give the output for the input stream 0 1 1 0 1 1 1.
2. A loop's branch is taken 999 times then not taken once at the end. A predictor always guesses "taken". How many mispredictions occur, and what is the total penalty if each costs 15 cycles?
3. Explain why a GPU is good at brightening every pixel of an image but poor at walking through a linked list.

Worked answers

Only look once you have had a proper go at each question.

1. $S0 \rightarrow (0) S0 \text{ out } 0; (1) S1 \text{ out } 0; (1) S1 \text{ out } 1; (0) S0 \text{ out } 0; (1) S1 \text{ out } 0; (1) S1 \text{ out } 1; (1) S1 \text{ out } 1$. Output: **0 0 1 0 0 1 1**.
2. Only the final not-taken outcome is mispredicted: **1 misprediction = 15 cycles** over 1000 branches, an average of only 0.015 cycles per branch.
3. Brightening applies the same simple operation independently to millions of pixels, so thousands of GPU cores can work at once. A linked list traversal is inherently sequential: you cannot find the next node until you have read the current one, so extra cores cannot help, and a CPU's fast single core and caches are better.